SHORT TERM PRODUCTION SCHEDULING OF AN AUTOMATED MANUFACTURING FACILITY

Stanley B. Gershwin, Ramakrishna Akella, Yong Choong, and Sanjoy K. Mitter

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, Massachusetts 02139

ABSTRACT

We describe extensions to the on-line hierarchical scheduling scheme for flexible manufacturing systems of Kimemia and Gershwin. Major improvements to all levels of the algorithm are reported, including algorithm simplification, substantial reductions of off-line and on-line computation time, and improvement of performance. Simulation results based on a detailed model of an IBM printed circuit card assembly facility are presented.

1. INTRODUCTION

This paper describes extensions to the work reported by Kimemia (1982) and Kimemia and Gershwin (1983) on the on-line scheduling of flexible manufacturing systems. Major improvements to all levels of the hierarchical algorithm are reported and simulation results are presented. The results indicate that the approach is practical, well-behaved, and robust. A full description of the results appears in Gershwin, Akella, and Choong (1984) and Akella, Choong, and Gershwin (1984).

A flexible manufacturing system (FMS) is one in which a family of related parts can be made simultaneously. It consists of a set of computer-controlled machines and transportation elements. The changeover time between different operations at a machine is small compared with operation times.

Processing a mix of parts makes it possible to utilize the machines more fully than otherwise. This is because different parts spend different amounts of time at the machines. Each part type may use some machines heavily and others very little or not at all. If complementary part types are selected for simultaneous production, the machines that are lightly used by some parts can be loaded with others that do require them.

In principle, therefore, line balancing can keep several machines busy at the same time. However, scheduling such a

system is difficult because there are several machines, several part types, and many parts. In addition, like all manufacturing systems, a FMS is subject to random disturbances in the form of machine failures and repairs, material unavailability, "hot" items or batches, and other phenomena. These effects further complicate an already difficult optimization problem.

Gershwin, Akella, and Choong present a hierarchical description of the manufacturing scheduling problem. At the top of the hierarchy are the long term decisions, such as what capital equipment to acquire. At the bottom is the decision of which part to load into an existing FMS, and when to load it. This is an extension to Kimemia and Gershwin's short term hierarchy (Figure 1).

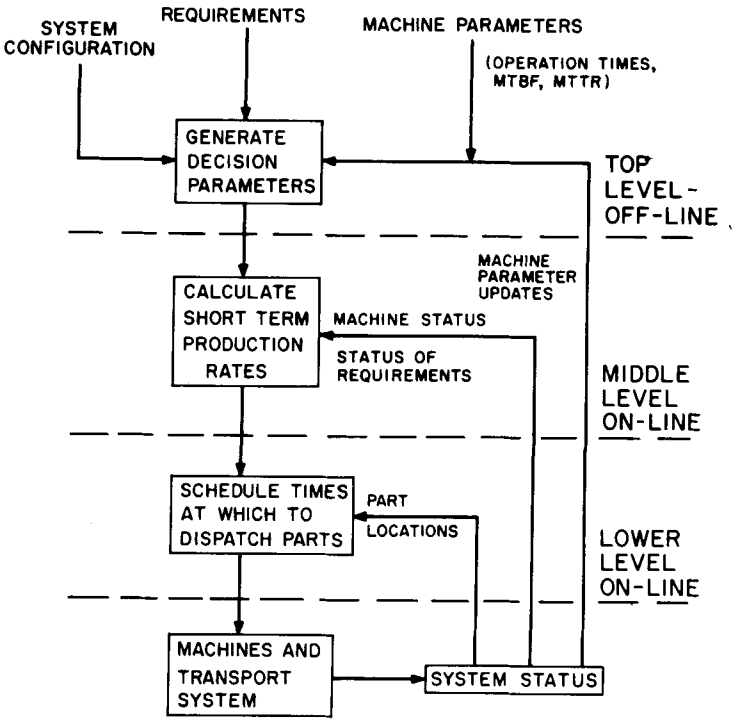

Figure 1. Hierarchical Production Algorithm.

2. CONTINUOUS FORMULATION

The purpose of the short-term FMS scheduling algorithm is to solve the following problem: when should parts (whose operation times at machines are on the order of seconds or minutes) be dispatched into an FMS whose machines are unreliable (with mean times between failures and mean times to repair on the order of hours) to satisfy production requirements that are specified for a week? Kimemia and Gershwin's approach decomposed the problem into two parts: a continuous dynamic programming problem to determine the instantaneous production rates and a combinatorial algorithm to determine the dispatch times.

The continuous part is further divided into the top and middle levels. The top level calculates a value or cost-to-go function and is executed off-line. The middle level uses the cost-to-go function to determine instantaneous flow rates and part mixes.

Assume that the production requirements are stated in the form of a demand rate vector d(t). Let the instanteous production rate vector be denoted u(t). Define x(t) to be production surplus. It is the cumulative difference between production and demand and satisfies

$$\frac{dx}{dt} = u(t) - d(t). \qquad (1)$$

If x(t) is positive, more material has been produced than is currently required. This surplus or safety stock is helpful to insure that material is always available over the planning horizon. However, it has a cost. Expensive floor space and material handling systems must be devoted to storage. In addition, working capital has been expended in the acquisition and processing of stored materials. This capital is not recovered until the processing is complete and the inventory is sold.

If x(t) is negative, there is a backlog, which is also costly. Backlog represents either starved machines downstream or unsatisfied customers. In the former case, valuable capital is underutilized; in the latter, sales and good will may be lost.

The production rate vector u is limited by the capabilities of the machines. Let part type j require time τ_{ij} on machine i for all of its operations. (Note that the order in which parts go to machines is not relevant for this calculation. Nor is the number of times a part visits a machine. For simplicity, we assume here that there is only one path for each part.) Then

$$\sum_j \tau_{ij} u_j(t) \leq \alpha_i(t) \qquad (2a)$$

where $\alpha_i(t)$ is 1 if machine i is operational and 0 if it is

down. More generally, if there is a set of identical type i machines, $\alpha_i(t)$ is the number of these that are operational at time t. Note also that

$$u_j \geq 0. \tag{2b}$$

Inequalities (2a) and (2b) can also be written as

$$u(t) \in \Omega(\alpha(t)). \tag{2}$$

3. TOP-LEVEL COMPUTATION (GENERATE DECISION TABLES)

Costs are incurred when x is far from zero. Kimemia and Gershwin describe the following dynamic optimization problem:

$$\begin{aligned} &\text{minimize } E \int g(x(t))\, dt \\ &\text{subject to (1), (2),} \\ &\text{and initial conditions } x(0) \text{ and } \alpha(0). \end{aligned} \tag{3}$$

The optimal value of the cost of this problem is called $J(x(0), \alpha(0))$.

Kimemia and Gershwin suggest a decomposition by which the n'th order Bellman partial differential equation for $J(x, \alpha)$ is replaced by n first order Bellman ordinary differential equations (where n is the number of part types, ie, the dimensionality of x, u, and d).

Kimemia further suggests approximating the solution to each one-dimensional dynamic programming problem with a quadratic cost function. Not only does this reduce data requirements, but it also simplifies the middle-level computation. As a result, the cost function is then written

$$J(x, \alpha) = \tfrac{1}{2}\, x^T A(\alpha)\, x + b(\alpha)^T x + c(\alpha) \tag{5}$$

where $A(\alpha)$ is a diagonal matrix, $b(\alpha)$ is a vector, and $c(\alpha)$ is a scalar whose value is not important. In this section, we propose an alternate technique for obtaining approximate values of the coefficients of (5).

The function $J(x(t), \alpha)$ is a decreasing function of t when α remains constant. The hedging point, given by

$$H_i(\alpha) = -b_i(\alpha)/A_{ii}(\alpha) \tag{6}$$

is the minimum value of $J(x, \alpha)$ for α fixed. It is the value that x reaches if α stays constant for a long time and if d is feasible, ie if $d \in \Omega(\alpha)$.

In order to calculate the hedging point, consider Figure 2 which demonstrates a typical trajectory of $x_i(t)$. Assume x_i has reached $H_i(\alpha)$, the hedging point corresponding to the machine state before the failure. Then u_i is chosen to be d_i

and x_i remains constant.

A failure occurs at time t_0 that forces u_i to be 0. This causes x_i to decrease at rate $-d_i$. In fact, if the failure lasts for a length of time T_r, then the minimum value of x_i is

$$H_i - d_i T_r. \tag{7}$$

Just after the repair (at time $t_0 + T_r$), u_i is assigned the value U_i. Assuming that this value is greater than demand d_i, x_i increases at rate $U_i - d_i$ until it reaches the hedging point H_i (at time t_3). At that time, u_i resumes its old value of d_i and x_i stays constant until the next failure, at time $t_0 + T_r + T_f$.

To simplify the analysis, we make several assumptions:

1. u_i is constant between the repair ($t_0 + T_r$) and when x_i reaches H_i (t_3).

2. T_r and T_f can be replaced by their expected values, the MTTR and MTBF.

3. The cost function $g(\ )$ in (3) penalizes positive areas in Figure 2 with weight a and negative areas with weight b, where a and b are positive scalars.

The positive area between t_0 and $t_0 + T_r + T_f$ is the area between t_0 and t_1 plus the area between t_2 and $t_0 + T_r + T_f$, where

$$t_1 = t_0 + H_i / d_i,$$

$$t_2 = t_0 + T_r - (H_i - d_i T_r) / (U_i - d_i)$$

and

$$t_3 = t_0 + T_r + d_i T_r / (U_i - d_i).$$

The positive area is

$$PA = \frac{1}{2} \frac{H_i^2 U_i}{d_i (U_i - d_i)} + H_i T_f - \frac{d_i T_r}{U_i - d_i}.$$

The absolute value of the negative area is

$$NA = \frac{1}{2} \frac{(H_i - d_i T_r)^2 U_i}{d_i (U_i - d_i)}.$$

Both terms are always positive.

The cost function, according to assumption 3, is then

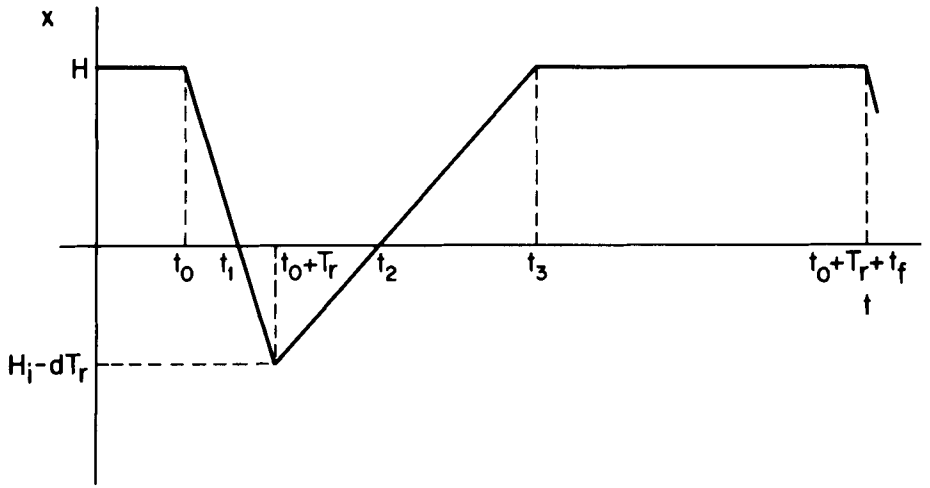

Figure 2. Typical x Trajectory During a Repair-Failure Cycle.

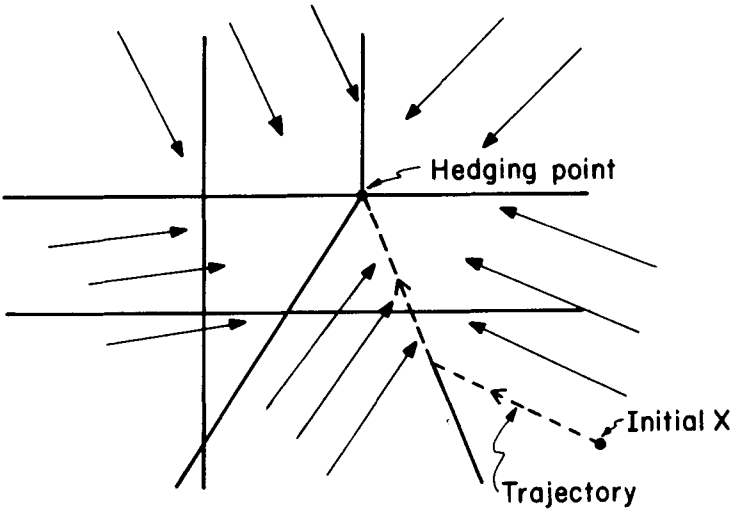

Figure 3. Regions of x-Space, and a Trajectory.

a PA + b NA

This quantity is the cost incurred per average repair and failure cycle of a machine. To find H_i, we must minimize it. This is not difficult because the cost is quadratic in H.

The minimizing H_i is

$$\frac{T_r\, d_i\, (b\, U_i - a\, d_i) - T_f\, a\, d_i\, (U_i - d_i)}{(a + b)\, U_i} . \qquad (8)$$

For machine states in which the demand is not feasible, this approach does not apply. The hedging point for such states must be larger than (8).

This calculation does not consider the fact that the failure may occur before the state reaches the hedging point or that the repair may occur before the state becomes negative. It assumes a specific form of g. These considerations and others should be the subject of future research, but it is important to observe that this method produced very satisfactory results.

$A_{ii}(\alpha)$ must be positive in order for J to be convex. Its value reflects the relative priority of part type i. Parts that have great value, or that would cause great difficulty if backlogged, or that pass through relatively unreliable machines should have large values of A_{ii}.

4.MIDDLE LEVEL (CALCULATE SHORT TERM PRODUCTION RATES)

Chattering

The optimal production rate vector u(t) satisfies the following linear programming problem.

$$\text{minimize} \quad \frac{\partial J(x,\alpha)}{\partial x}\, u \qquad (9)$$

subject to (2).

This is a feedback law since the problem is only specified when x and α are determined. The numerical solution of (9) is implemented on-line at the middle level of the hierarchical algorithm.

For every α, x-space is divided into a set of regions (open, connected sets) and the boundaries between them. Each region is associated with a corner of $\Omega(\alpha)$. When x is in the interior of region R_i, the value of u that satisfies (9) is the corresponding corner P_i.

Kimemia and Gershwin implemented (9) in a simulation by solving it every time step (one minute). This worked well while x was in the interior of a region R_i. However, when x crossed certain boundaries between regions, this approach worked poorly. After x(t) crossed such an attractive boundary, the value of u corresponding to the new region R_j was such that the derivative (1) pointed toward R_i. When x(t) crossed the boundary back into R_i, the derivative pointed again to R_j. Thus, u(t) jumped between adjacent corners P_i and P_j of $\Omega(\alpha)$.

This chattering behavior is undesirable. It allows the flow rate to change more frequently than parts are loaded into the system. The flow rates are such that, very often, at least one of the machines is fully utilized or totally unutilized (since u(t) is at an extreme point of $\Omega(\alpha(t))$. When u jumps frequently from one corner of $\Omega(\alpha)$ to another, the algorithm is trying to switch rapidly from keeping one set of machines fully loaded or unloaded to keeping another fully loaded or unloaded.

It cannot do this successfully: neither set of machines is fully loaded or unloaded. As a result, if the demands on the system are near its capacity, it will fail to meet the demands. This behavior was observed by Kimemia (1982).

Planar Boundaries

Kimemia shows that the optimal $J(x,\alpha)$ is convex in x for each α. He also shows that J decreases when u satisfies (9) and d is feasible. The minimal value of J is achieved when x is at the hedging point. When J is given by (5), its minimum is reached at (6). Gershwin, Akella, and Choong show that when J is quadratic, the regions of x-space (in which the solution u of (9) is constant at a corner of $\Omega(\alpha)$) are cones.

Linear program (9) can now be written

$$\begin{aligned} &\text{minimize} && c(x)^T u \\ &\text{subject to} && D\,u = e \\ & && u \geq 0 \end{aligned} \qquad (10)$$

where u has been expanded with slack variables so that inequality constraint (2a) can be written as an equality, and

$$c(x) = A\,x + b.$$

(Note that arguments α and t are suppressed.) The standard solution of (10) (Luenberger, 1977) breaks u into basic (u_B) and non-basic (u_N) parts, with c(x) and D broken up correspondingly. The basic part of D is a square, invertible matrix. By using the equality in (10), u_B can be eliminated, and the problem becomes

$$\text{minimize } c_R(x)^T u_N \qquad (11)$$

$$\text{subject to } u_N \geq 0$$

where the constraint on u_B has been suppressed, and where

$$c_R(x)^T = c_N(x)^T - c_B(x)^T D_B^{-1} D_N$$

is the reduced cost. If all components of c_R are positive, then there is a solution to (11): $u_N = 0$. This and the corresponding u_B form an optimal solution to (10). Otherwise, (11) does not have a bounded optimal solution and (11) is not equivalent to (10).

It is important to note that since c is a function of x, the basic/non-basic breakup of this problem depends on x. That is, the set of components of u that are treated as basic varies as a function of x.

At every x in region R_i, corner P_i is the optimal value of u for (10). In each region, then, there must be a basic/non-basic break-up of (10) which is constant. Consequently, $c_R(x)$ must be positive everywhere in its own region and it must have some negative components elsewhere. The boundaries of the regions are determined by some components of $c_R(x)$ being equal to zero.

The boundaries of the regions are therefore portions of hyperplanes. This is because c(x) is linear in x. Consequently $c_N(x)^T$ and $c_B(x)^T$ and therefore $c_R(x)$ are also linear in x.

Qualitative Behavior of Trajectories

Since u is constant throughout a region, dx/dt is also constant. The buffer state x travels along a straight line in the interior of each region. As indicated in Figure 3, such lines may intersect with one or more boundaries of the region. When x reaches a boundary, u and therefore dx/dt changes.

Some boundaries are such that when they are reached, the trajectory continues, after changing direction, into the adjacent region. Others, that we call attractive boundaries, are different. The trajectories on both sides of such boundaries point toward them. Consequently, the trajectories tend to move along the boundaries.

We can now qualitatively describe the trajectory. After a machine state change, x(t) is almost always in the interior of a region. It moves in the characteristic direction of that region (which corresponds to a corner of the $\Omega(\alpha(t))$ polyhedron) until it reaches a boundary. If the boundary is not attractive, x(t) moves in the interior of the next region until it reaches the next boundary. The production rate vector u jumps to an adjacent corner. This behavior continues until x(t) encounters an attrac-

tive boundary. At this time, the trajectory begins to move along the boundary and u(t) jumps to a point on the edge of $\Omega(\alpha)$ between the corners corresponding to the regions on either side of the boundary.

The trajectory continues until it hits the next attractive boundary. After that, x(t) moves along the intersection of three regions. The production rate vector is on the surface determined by the three corners corresponding to these regions.

This behavior continues: x(t) moves to lower dimensional boundaries and u(t) jumps to higher dimensional faces. It stops when either the machine state changes (that is, a repair or failure takes place) or u(t) becomes constant. If the demand is feasible (that is, if $d \in \Omega(\alpha)$) then the constant value for u is d. When that happens, x also becomes constant and its value is the hedging point. If the demand is not feasible, x does not become constant. Instead, some or all of its components decrease without limit.

Consequently, for a constant machine state, the future behavior of x(t) would be determined from its current value. We call this the "conditional future trajectory" or the "projected trajectory".

4.2 Calculation of the Conditional Future Trajectory

Assume that the conditional future trajectory is to be calculated at time t_0. This may be due to a machine state change.

As soon as the machine state change occurs, linear program (10) is solved. Thus the basic/non-basic split is determined and the $c_R(x)$ function is known.

The production rate vector at $t=t_0$ is denoted u_0. The production rate remains constant at this value until $t=t_1$, which is to be determined. In $[t_0, t_1]$, x is given by

$$x(t) = x(t_0) + (u_0 - d)(t - t_0)$$

where $x(t_1)$ is on a boundary. Then t_1 is the smallest value of t for which some component of $c_R(x(t))$ is zero. It is easy to calculate this quantity since c_R is linear in x and x is linear in t. Once t_1 is found, $x(t_1)$ is known. Define $h(x(t))$ to be the component of $c_R(x(t))$ that reaches zero at $t=t_1$. Because h is a linear scalar function of x, we can write

$$h(x(t)) = f^T (x(t) - x(t_1)).$$

For $t > t_1$, there are two possibilities. The trajectory may enter the neighboring region and travel in the interior until it reaches the next boundary. Alternatively, it may move along the boundary it has just reached. To determine whether or not the boundary is attractive, we must consider the behavior of

h(x(t)) in its neighborhood.

We know that h(x) is negative in the region across the boundary since this is how the regions are defined. We must determine whether h is increasing or decreasing on trajectories inside that region. If h is decreasing, x moves away from the boundary (where h is zero) into the interior. If h is increasing, trajectories move toward the boundary which must therefore be attractive.

One value of x which is just across the boundary is

$$x'' = x(t_0) + (u_0 - d)(t_1 + \epsilon - t_0)$$

$$= x(t_1) + (u_0 - d)\epsilon.$$

This is the value x would have if u were allowed to be u_0 until $t_1 + \epsilon$.

Let u" be the solution to (10) in the adjacent region. That is, (10) is solved with x given by x". (This can be performed efficiently.) Let x^* be the value of x at $t_1+\epsilon$ if u" were used after t_1. That is,

$$x^* = x(t_1) + (u'' - d)\epsilon.$$

Then

$$h(x'') = f^T (u'' - d)\epsilon.$$

Therefore h is increasing and the boundary is attractive if and only if

$$f^T (u'' - d) > 0.$$

If the boundary is not attractive, define $u_1 = u''$. Then the process is repeated to find t_2, $x(t_2)$, t_3, $x(t_3)$, and so forth until an attractive boundary is encountered. (It should be remembered that this is an on-line computation that is taking place at time t_0. The future trajectory is being planned.)

If the boundary is attractive, a value of u must be determined which will keep the trajectory on it. Otherwise chattering will occur. For the trajectory to stay on the boundary,

$$h(x(t)) = 0$$

or, since $h(x(t_1)) = 0$,

$$\frac{d}{dt} h(x(t)) = f^T(u - d) = 0. \qquad (12)$$

Although u is an optimal solution to (10), it is no longer

determined by this linear program. In fact u_0, u", and any convex combination of them are optimal. This is because one or more of the reduced costs is zero while x is on a boundary. Consequently, the new scalar condition (12) is required to determine the solution. The linear program is modified as follows:

$$\begin{aligned} &\text{minimize} && c(x)^T\, u \\ &\text{subject to} && D\, u = e \\ & && u \geq 0 \\ & && f^T u = f^T d \end{aligned} \qquad (13)$$

By adding equation (12) to (13), we are requiring that the solution keeps x(t) on the boundary. We are also replacing the reduced cost which has become zero with a new equation, so that the new problem has a unique solution.

The solution to (13) is the value of u that keeps the trajectory on the boundary. As before, this value is maintained until a new boundary is encountered.

New boundaries may still be attractive or unattractive. The same tests are performed: x is allowed to move slightly into the next region to determine the value of u. The time derivative of the component of the reduced cost that first reaches zero (h) is examined. If it is negative, the boundary is unattractive and the trajectory enters the new region. If it is positive, a new constraint is added to linear program (13).

Constraints, when added to (13), are not deleted. As the number of constraints increases, the surfaces that u is found on in $\Omega(\alpha)$) increase in dimension. That is, u is first on a corner. When the fist attractive boundary is encountered, u is on the edge formed by the convex combination of the corners corresponding to the regions adjacent to the boundary. When the next attractive boundary is reached, u is a convex combination of three corners, and so forth.

At the same time, x is found in regions of decreasing dimension. After a machine state change, $x(t_0)$ is in the interior of a region of full dimensionality. The first attractive boundary x(t) reaches is a hyperplane separating regions of full dimensionality, so its dimensionality is one less than full. The next boundary is the intersection of two such boundaries and thus has dimensionality one smaller.

Since this is a finite dimensional system, this process must terminate. There are two cases. If the demand is feasible, ie if d is a feasible solution of (10), then d is a feasible solution of (13). This is because d satisfies (12) for all f. As new constaints of the form (12) are added to (13), d remains feasible. Finally, if enough linearly independent constraints are added, there is only a single feasible solution to (13) and that

is u=d.

Since the dynamics of x are given by (1), x remains constant when u=d. The value of this constant is the hedging point, discussed above, which is the minimum of $J(x, \alpha)$ for the current value of α.

If the demand is not feasible, u cannot be equal to d and thus x cannot become constant. Instead, the process described above terminates with u satisfying linear program (13) including one or more constraints of the form (12). The vector x(t) is eventually of the form

$$x(t) = x(t_j) + (u - d)\, t.$$

Since d is not feasible, some or all of the components of u - d are negative. The corresponding components of x decrease without limit.

4.3 COMPUTATIONAL CONSIDERATIONS

The conditional future trajectory is calculated whenever the machine state changes, either due to a failure or a repair. It may also be calculated under other conditions: periodically, to ensure that the actual trajectory is close to the projected trajectory; or after unanticipated events such as parts not being loaded into the system in the prescribed manner.

To begin the computation, a linear programming problem (10) must be solved. The number of variables (production rates and slack variables) is the number of part types plus the number of distinct machines.

As each boundary is reached, one (if unattractive) or two (if attractive) additional programs are solved. The numerical effort is very small, however, since each starting basic feasible solution is the solution of the previous problem. We expect that no more than a few pivots of the simplex method will be required to find each new solution.

5. LOWER LEVEL

Lower Level (SCHEDULE TIMES AT WHICH TO DISPATCH PARTS)

The new part loading scheme is based on the conditional future trajectory (x(s), s>t). Define the actual surplus of part type i at time t to be

$$x^A_i(t) = [\text{number of parts of type } i \text{ loaded during } [0,t]] - d_i\, t.$$

Note that $x^A_i(t)$ is an irregular sawtooth function of time. It jumps by 1 each time a part is loaded. At other times,

it decreases at rate d_i.

The loading strategy insures that $x^A(t)$ is near $x(t)$. The strategy is: at each time step t, load a part of type i if

$$x^A_i(t) < x_i(t). \qquad (14)$$

Do not load a part of type i otherwise. A rule is required to resolve conflicts; it probably does not matter what that rule is since conflicts will not arise very often.

Behavior of the New Strategy

Figure 4 demonstrates the behaviors of projected and actual trajectories. It shows a portion of a projected trajectory and of an actual trajectory that was determined by this method.

The actual trajectory remains as close as possible to the projected trajectory. (There are five other such trajectories because six part types are being produced in the simulated system). No difficulty is experienced at the time (about 9740) when the loading rate changes.

6. SIMULATION RESULTS

A detailed simulation of an IBM flexible manufacturing system was written to test the hierarchical scheduling policy and to compare it with other reasonable ploicies. The simulation is described in Akella, Bevans, and Choong (1984). A full description of the results appears in Akella, Choong, and Gershwin (1984).

Six part types are being made simultaneously. Failures and repairs of machines take place at random times. Each MTBF is 10 hours and each MTTR is 1 hour, so that the efficiencies are all 91%. Demands are chosen so that the machines are utilized 98%, 91%, 96%, and 97% of the expected available time.

A variety of alternative policies was formulated to compare with the hierarchical policy. Policy X was: If more than N parts are in the system, do not load a part. If N or fewer parts are in the system, load the part that is furthest behind or least ahead of demand. Do not allow any part type to get more than K parts ahead of demand.

Parameter N must be chosen. Little's law (Little, 1961) gives some guidance, but simulation experience indicates that behavior can depend critically on its value.

Figure 5 displays the results of four runs of the hierarchical policy and four runs of Policy X. All runs were performed with the same seed for the random number generator. That is, each had the same sequence of repairs and failures. The horizontal axis displays the average number of parts in the system. The

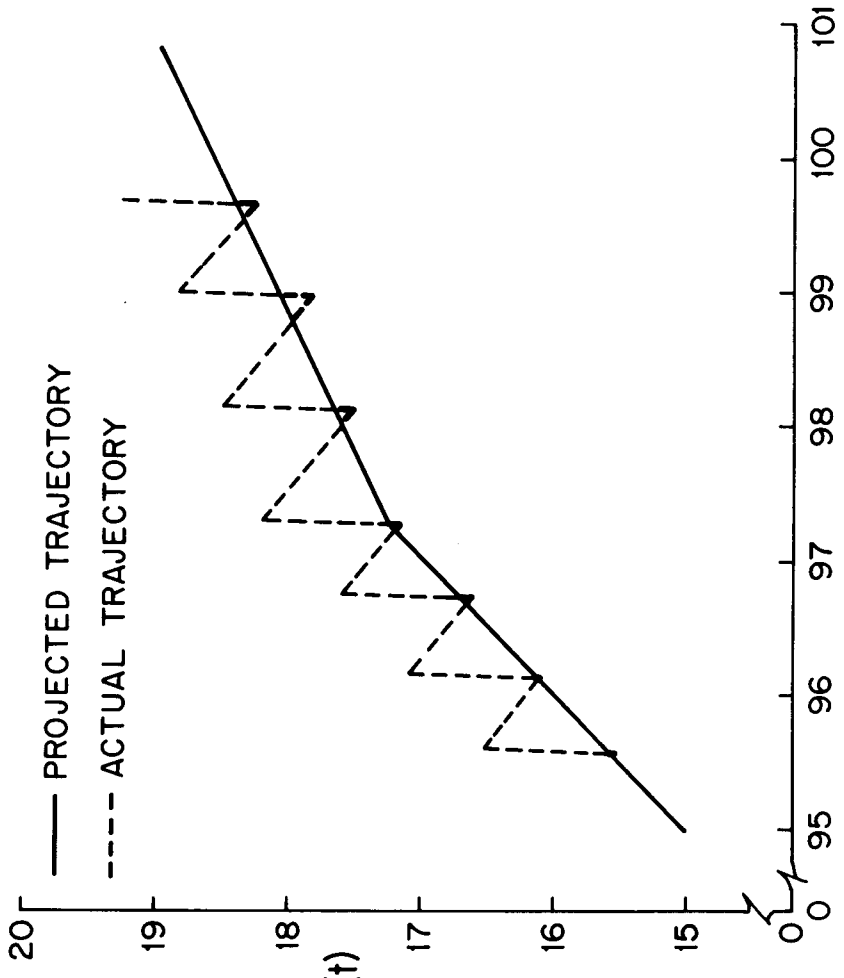

Figure 4. Lower Level Behavior.

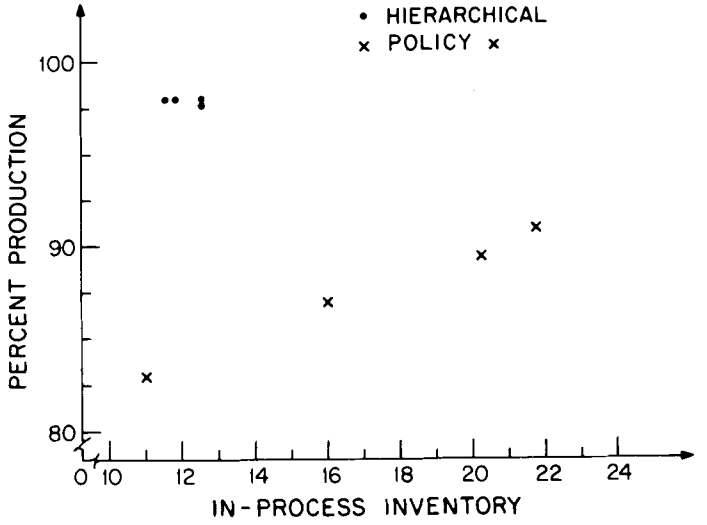

Figure 5. Production and Inventory Comparisons.

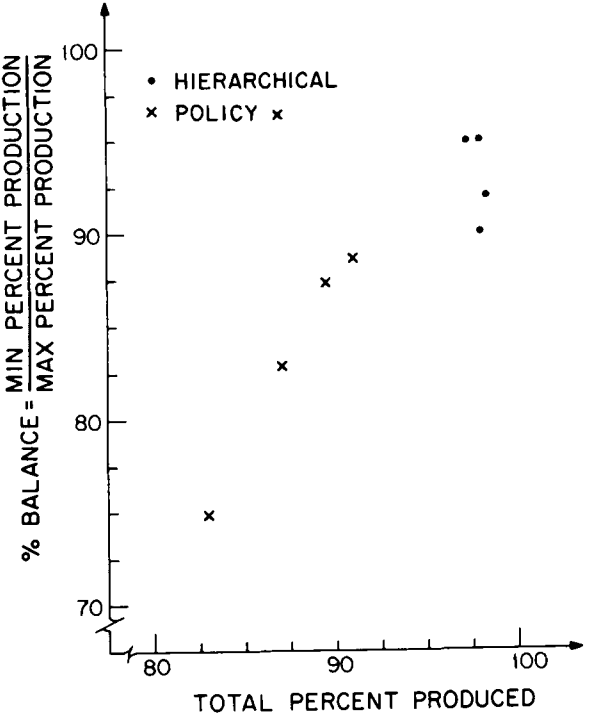

Figure 6. Balance Comparisons.

vertical axis shows the percentage of the total requirements that was actually produced.

The four hierarchical runs had different values of A and b. The Policy X runs had different values of N. Figure 5 indicates that the hierarchical strategy produced superior results. The in-process inventory was lower and the production percentage was greater: in fact, over 98%. In addition, although the values of the A and b parameters differed considerably, the four hierarchical points are clustered quite close together. This indicates that the policy is not sensitive to these parameters.

Figure 6 demonstrates how the policies satisfy balance requirements. The horizontal axis is the same as the vertical axis of Figure 5. To define balance, let

$$Z_i = \frac{\text{number of type i parts produced during the run}}{\text{number of type i parts required during the run}} .$$

Then balance is defined as

$$\frac{\min Z_i}{\max Z_i} .$$

It is important that balance be near 100% to ensure that only what is required is produced.

Akella, Choong, and Gershwin (1984) present a fuller comparison of these and other policies. The more sophisticated policies performed better than Policy X, but not as well as the hierarchical policy.

7. SUMMARY AND CONCLUSIONS

The hierarchical scheduling policy devised by Kimemia and Gershwin for flexible manufacturing systems has been further developed and tested. This policy is designed to respond to random disruptions of the production process. In its current formulation, it treats unpredictable changes in the operational states of the machines: repairs and failures. All levels of the policy have been improved, and the policy shows great promise for practical application.

REFERENCES

R. Akella, J. P. Bevans, and Y. Choong (1984), "Simulation of a Flexible Electronic Assembly System," Massachusetts Institute of Technology Laboratory for Information and Decision Systems Report to appear.

R. Akella, Y. Choong, and S. B. Gershwin (1984), "Performance of Hierarchical Production Scheduling Policy," Massachusetts Institute of Technology Laboratory for Information and Decision Systems Report LIDS-FR-1357.

S. B. Gershwin, R. Akella, and Y. C. Choong, "Short Term Production Scheduling of an Automated Manufacturing Facility," Massachusetts Institute of Technology Laboratory for Information and Decision Systems Report LIDS-FR-1356.

J. G. Kimemia (1982), "Hierarchical Control of Production in Flexible Manufacturing Systems, Massachusetts Institute of Technology Laboratory for Information and Decision Systems Report LIDS-TH-1215.

J. G. Kimemia and S. B. Gershwin (1983), "An Algorithm for the Computer Control of Production in Flexible Manufacturing Systems, IEE Transactions, Volume 15, No. 4, December, 1983, pp. 353-362.

D. Luenberger (1977) Introduction to Linear and Nonlinear Programming, Addison-Wesley.

J. D. C. Little (1961) "A Proof for the Queuing Formula: $L = \lambda W$," Operations Research, Volume 9, Number 3, pp. 383-387.

ACKNOWLEDGMENTS

Research support has been provided by the Manufacturing Research Center of the Thomas J. Watson Research Center of the International Business Machines Corporation; and by the U. S. Army Human Engineering Laboratory under Contract DAAK11-82-K-0018.