International Journal of Flexible Automation and Integrated Manufacturing, 3(3&4): 301-316 (1995)

Hierarchical Scheduling of Robotic Assembly Operations in a Flexible Manufacturing System

Elizeth G. Araujo,[1] Gordon A. Dakin,[2] Manfred Huber,[1] and Roderic A. Grupen[1*]

[1]Laboratory for Perceptual Robotics, Department of Computer Science, University of Massachusetts, Amherst, Massachusetts 01003; [2]Crystaliz, Inc., 695 Virginia Road, Concord, Massachusetts 01742

[*] To whom all correspondence should be addressed. More information is available at http://piglet.cs.umass.edu:4321/.

ABSTRACT

An approach to hierarchical scheduling for assembly in a fully automated flexible manufacturing system is presented. The three-level production system consists of a *master production plan*, which incorporates new customer requests and makes long-term production guarantees; a *production flow level*, which plans and monitors the flow of subassemblies among workstations; and a *scheduler level*, which sequences and dispatches assembly operations. An independent scheduler at each workstation coordinates robot motions and conveyor transferals to interleave the operations related to different subassembly products. The higher priorities of tardy subassemblies are weighed against immediate opportunities to exploit the current configuration of the workstation and maximize overall throughput. We focus on the scheduler level, wherein heuristics and combinatorial search are used to exploit each workstation's capacity optimally. Two scheduling algorithms are described — a *fine-granularity* scheduler, which analyzes robot-conveyor interactions with precise timing, and an *anytime* scheduler, which stochastically generates and evaluates alternative schedules until the tradeoff between expected plan improvements and dispatching delays is no longer favorable.

KEYWORDS: flexible manufacturing, automation, hierarchical production model, value-based scheduling, anytime algorithm.

I. INTRODUCTION

Most manufactured goods in the United States are produced in small batch processes, which places pressure on manufacturing facilities to reduce time to market and to improve response to consumers. At the same time, advances in technology allow for increased automation, even under these conditions.

©1995 by Begell House, Inc.

Flexible, low-quantity automated manufacturing poses different challenges than does traditional high-volume production. While mass production may dedicate an entire assembly line to a single product, small batch production requires rapid job switching to reduce idle time and production costs. Such a flexible manufacturing system (FMS) consists of a group of machines, each capable of performing a variety of operations and all integrated by an automatic transport facility for transferring parts. Automatic tool changers and built-in sensors enable machines to switch between different operations, building a suite of products and subassemblies that may contain identical materials and subassemblies. This resource sharing requires versatile manufacturing facilities to cope with uncertainties in part feeding and the different requirements of each product. However, this flexibility, plus the required use of sensory input, leads to larger variances in task execution times and nondeterministic arrival times for parts and subassemblies. To control such a system, the planning and scheduling level has to accommodate these uncertainties as well as the resource constraints involved in coordinating multiple workstations.

The number of items produced by an FMS is dictated by a master production plan, which specifies the demand for each product within a given time period. The production management involved in realizing the master plan involves decisions from the production-flow level (production rates and parts routing) to the machine level (job dispatching and material transport). Much of the research in FMS scheduling has focused on part routing (machine selection and route splits) and job sequencing (the orderly dispatch of parts to each machine).

Job-to-machine assignments are mainly based on required production rates, operation precedences, machine suitability, and workload balancing. Approaches to solving the routing problem include *integer programming*,[1, 2] *queuing theory*,[3] and *control theory*.[4, 5, 6] While the integer programming yields optimal results, it is unwieldy when a large number of machines is involved. The more tractable queuing theory approach models job arrivals and processing times as probability distributions and permits a rapid evaluation of system performance. It fails, however, to handle occasional disruptions, such as machine failures, which are easier to represent in a control theoretic model, where production flow is modeled as a continuous dynamic programming problem.

Policies for sequencing and dispatching jobs are normally based on the due dates or flow rates of the processed items and the capacity and suitability of the machine.[7] Simple priority rules, such as *shortest imminent job* or *critical ratio*,[8] may be evaluated for effectiveness through FMS simulation[9] and refined through perturbation analysis.[10] Artificial intelligence approaches, which rely on powerful rules of thumb acquired from human experts, have also been explored.[11] As with the routing problem, optimal solutions for the sequencing problem are intractable for a larger FMS.

Overall, the job-shop problem, in which M operations are optimally scheduled on N machines, is known to be NP-hard when $M > 2$.[12] To cope with this, many methods used statistical approximations or powerful heuristics to obtain tractable solutions. In this context, the computational advantages of a hierarchical control model can be demonstrated.[4, 6] Another important issue is the ability of the control system to cope with variations in execution times and nondeterministic part arrivals, which severely compromise the quality of

a priori plans of operation sequences. While statistical treatment of these actors might lead to acceptable solutions, idle times cannot be avoided without dynamic changes in routing and operation sequences.

The approach presented in this paper employs a hierarchical model with a distributed scheduling level to optimize machine utilization in the presence of uncertainties. This decomposition also makes it possible to integrate a large number of workstations. The next section presents the overall approach, and the remainder of the paper focuses on scheduling at the workstation level and explores planning methodologies for operation scheduling. Because the experimental workcell is equipped with storage buffers, it involves part transferal issues related to linear and circular assembly line scheduling,[13, 14] as well as assembly and internal resource sharing.

II. THE HIERARCHICAL PRODUCTION MODEL

To cope with the computational complexity of a large FMS and uncertain operational timing, the approach presented in this paper employs a hierarchical production scheduling model. Different parts of the planning are distributed among different levels and individual workstations. The production management system shown in Figure 1 comprises three levels: *master production*, *production flow*, and *scheduler*.

At the top level, a master production plan specifies the set of products to be manufactured for each work period (e.g., workday). At this level, consumer requests are incorporated whenever worst-case production-time estimates, material availability, and delivery times allow their completion before the due date. This allows the system to give immediate feedback if timely delivery of a new request can be guaranteed and to issue orders to suppliers without delay. No actual sequencing or routing decisions for the requests are made at this level, only a conservative capacity analysis for the system. The master production plan may also incorporate expected machine down-time and maintenance.

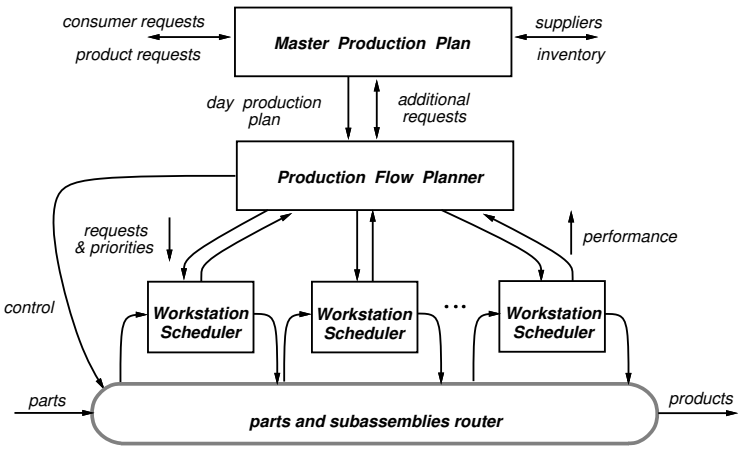

FIGURE 1
Hierarchical Production Model

At the beginning of each work period, the production-flow planner evenly distributes the subassembly requests necessary for the products of that period's master production plan among the request buffers of suitable workstations, and it sends material requests to the storage area. Throughout the production process, this level has three main coordinating tasks based on performance feedback from the lower scheduling level: controlling part routing, monitoring and adjusting flow rates, and assigning requests from master production plans of future work periods if enough time was reclaimed on the lower level. Again, no request sequencing is performed at this level; all decisions are based on conservative capacity estimates, performance feedback from the individual workstations, and the state of the routing system.

Finally, the scheduler level is distributed among the workcells. Each workstation opportunistically decides in which order to execute the operations required to build the subassemblies of its request buffer, given the currently available parts and the values (i.e., priorities) of the requests. Multiple assembly sites within one workcell allow the scheduler to interleave assembly operations for multiple subassemblies, in order to optimize machine utilization and accumulated value. Throughout production, each workcell feeds back information about its current performance, subassemblies produced, and amount of time reclaimed with respect to the worst-case production-time estimates used for the guarantees in the master production plan. To account for uncertainties and the opportunistic character of operation scheduling at the workstation level, all levels have to continuously update their plans to utilize reclaimed time and performance feedback. The rate at which such feedback is passed on and incorporated at each level, however, decreases drastically with each step up the hierarchy. This allows the whole system to give long-term guarantees while remaining responsive to timing uncertainties.

II.1. Priority Scheduling and Production Flow

In the hierarchical model presented, the actual production sequence is determined in a distributed fashion at the workstation level. Each workcell schedules its operations opportunistically to optimize the accumulated value, based on the content of its request buffer and the materials currently available to the workstation. Therefore, the value V of each subassembly directly influences the production flow of the whole system, which allows the production flow level to coordinate the system by selecting initial values and modifying them online to obtain the desired production rates for each subassembly.

Whenever a new subassembly request, j, is passed down to the request queue of a workstation, the production flow planner attaches to it a base priority, P_j, based on the importance or price of the corresponding product. To represent temporal urgency more explicitly, this priority is augmented with an artificial "soft" deadline, t_j, to yield a time-varying product utility $U_j(t)$:

$$U_j(t) \;=\; \begin{cases} P_j & if\ t \le t_j \\ P_j - m_j(t - t_j) & if\ 0 < m_j(t - t_j) < P_j \\ 0 & otherwise \end{cases} \qquad (1)$$

where m_j is the rate of decrease in product utility after the deadline.

Because most of the workstations produce subassemblies that will be components of higher-order subassemblies produced downstream, it is important to actively monitor and enforce desired flow rates to avoid the accumulation of work in progress and lags in production. To achieve this, the production flow level determines desired production rates for the various subassemblies. It then constantly monitors the production, using performance feedback from the workstation level, and adjusts the scheduling values, V_j, to

$$V_j(t) \;=\; U_j \;+\; S_k \tag{2}$$

where j is a request for a subassembly of type k, and S_k is a priority modification reflecting the discrepancy between actual and desired flow rates of this subassembly type. Thus, for example, V_j increases when production of the subassembly type k lags behind, which in turn increases subassembly k's production. The production flow controller thus enforces specified flow rates indirectly through negative feedback by adjusting request values V_j through the feedback terms S_k.

A second important function performed by the production flow planner is routing materials and subassemblies to the different workstations. Whenever parts enter the system or new subassemblies are produced, the flow control level decides which workstation to route them to based on the requirements of their requests and performance feedback from the individual workcell schedulers. Among all the workstations that need a part of this type, the one that has the least schedulable operation time is selected, to avoid idle times in the future. The part is routed there and kept in a local buffer accessible directly by the workstation.

Both mechanisms – part routing and scheduling value modification – allow the production flow planner to control the overall production of the system without the need to compute effective operation sequences for all workcells. This dramatically reduces the complexity of the decisions necessary, allowing for more workcells in the system. It also allows each workcell to opportunistically schedule its own requests to optimize value and utilization and thus to reclaim time at the lowest level.

The remainder of this paper focuses on the scheduler level, where assembly operations related to different subassembly products are interleaved to achieve near-optimal workstation capacity. Two schedulers have been implemented: *fine-granularity* and *anytime*.

III. VALUE-BASED SCHEDULING IN A ROBOTIC ASSEMBLY WORKSTATION

Scheduling in each workcell is critically important in this hierarchical production control model. At this level, local robot motions and conveyor transferals are coordinated to optimize resource utilization and, thus, throughput. Because of nondeterministic production and part arrival times, *a priori* schedules would be inefficient, requiring schedules to be created online based on currently available parts and subassembly values.

Planning optimal action schedules is an active research area in AI. The combinatorial character of this problem often renders exhaustive planning infeasible

because excessive planning times would lead to idle times in the system and thus to poor performance. It is therefore necessary to either restrict the task of the planner or accept suboptimal solutions. Changing the granularity of the plan is one way to change the complexity of the problem. While complexity increases when planning using fine-granularity actions, it decreases when using coarser schedules on the part or even product level. Coarse granularity, however, might require more than one level of planning, because high-level primitives cannot be executed directly.

Another way to avoid long idle times is to use *anytime algorithms*, which, by allowing specification of their execution time, provide a way to overcome time constraints by trading off computation time for plan quality based on performance expectations.

Following is a description of how these techniques were incorporated into different value-based schedulers and evaluated on an example workstation.

III.1. The Assembly Workstation

To test the scheduling level of the control hierarchy, the assembly workstation depicted in Figure 2 was built in hardware as well as in simulation.

This model contains a rotary arrival buffer, a linear transfer buffer, a hand-arm robot system, and an assembly table with four assembly sites. Raw parts and subassemblies completed by other workstations are deposited on the rotating arrival buffer. Motion commands are issued to align empty bins of the transfer buffer with those in the arrival buffer and to automatically transfer parts. These parts are located on the transfer buffer by a camera, allowing the robot

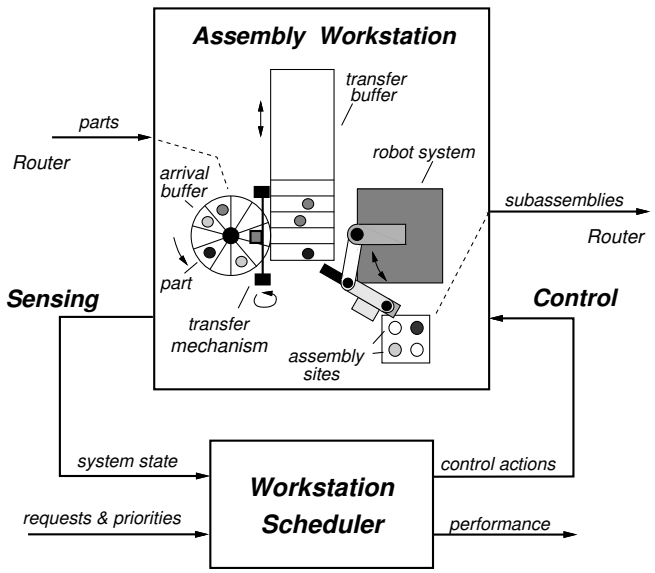

FIGURE 2
FMS workstation model.

to acquire and transport them to an assembly site, where assembly operations are performed. The arrival buffer, which can hold many parts, and the multiple assembly sites allow the system to plan which requests to build and to interleave them to react to new requests and changes in value. The role of the scheduler is to coordinate buffer movements, part transferals, visual sensing, and assembly operations to maximize the accumulated value of the produced subassemblies, according to the priority scheme described in the previous section. The schedule has to obey given precedence and mutual exclusion constraints among the operations. In the following sections, subassemblies (products) consist of three parts, requiring six operations for completion.

III.2. The Fine-Granularity Scheduler

The fine-granularity (FG) scheduler analyzes operation sequences by simulating buffer motions and robot-buffer interactions. Operation sequences are compared by Gantt-chart analysis, which is performed at a relatively fine level of temporal precision, corresponding to a fine granularity of planning. Two versions of the FG scheduler were implemented – "brute-force" and heuristic. The brute-force scheduler performs an unconstrained search among alternative operation orderings, optimizing the combined product values of completed products. Because the search time for an optimal ordering is exponential in the number of operations, the brute-force scheduler cannot plan many operations into the future, and it is discussed here only for comparison with the heuristic scheduler.

The heuristic scheduler contains two hierarchical planning levels: high-level product selection and low-level operation sequencing. At the high level, the scheduler selects a sequence of products to assemble, heuristically estimating each product's completion time and associated product value by assuming constant operation times and ignoring context dependencies at run-time. This coarse but efficient time analysis permits the heuristic scheduler to plan several products ahead. Lower-level planning then proceeds as in the brute-force scheduler, but only the operation sequences required to complete the first product in the chosen sequence are enumerated. Both scheduler versions replan their operation sequences after every assemble operation to incorporate new part arrivals, as well as new or modified product requests.

The technique used by the FG scheduler to evaluate operation sequences is demonstrated here by way of example. In the workstation shown in Figure 3, a type-A part occupies a bin in the arrival buffer, and a type-B part occupies a bin in the transfer buffer. The arrival buffer rotates at a rate of $T_r = 0.1$ time units per bin; part transferals to the transfer buffer require $T_t = 0.2$ time units; and robotic transport from the transfer buffer to the assembly table, plus subsequent assembly, takes $T_a = 0.8$ time units. There are currently two requests for subassemblies: (1) a product consisting of a type-C, a type-B, and a type-A part, initially valued at 30 points, and (2) a product consisting of three type-B parts, initially valued at 20 points. Soft deadlines for completing the two products occur at times 2.0 and 1.2, respectively. The first requested product is near completion in assembly site 2, lacking only a type-A part. In assembly site 1, the second requested product lacks only a type-B part. Exactly three operations may now be performed: *transfer A*, *assemble A*, and *assemble*

B. The planner must decide which sequence optimizes the combined values of the two completed products. The Gantt chart in Figure 3 reveals that the third sequence, which interleaves the two products' operations, yields the greatest sum (40) of product values. Although this sequence satisfies neither soft deadline, it balances the products' relative priorities optimally.

III.2.a. Performance of the FG scheduler. Whereas the heuristic scheduler requires comparatively little run-time and always finishes within the timespan of an assemble operation, the run-time of the brute-force version is roughly proportional to 3^n, where n is the number of planned operations, leading to a declining performance for $n > 6$. The brute-force scheduler was therefore operated offline, running the scheduler and workstation simulator nonconcurrently.

The FG schedulers were tested using a wide variety of part arrival sets and product requirements. Representative of typical scheduler performance, the graphs in Figure 4 show the incrementally increasing sums of product values achieved by the two schedulers. Figure 4(a) contrasts the behavior of the "greedy" brute-force scheduler with the heuristic scheduler's greater lookahead. The brute-force scheduler achieves greater short-term payoffs; the heuristic scheduler, with its four-product lookahead, leads to superior long-term performance. The brute-force scheduler achieves greater payoffs with a lookahead window of eight operations than with seven. (Brute-force scheduling experiments were not performed for lookahead windows larger than eight operations, because of CPU time constraints.)

Figure 4(b) shows scheduler performance for two part-arrival loads, using a constant set of product requests. For these experiments, the FMS workstation simulator restricted the number of possible transfer and assemble operations by delaying part arrivals appropriately. This figure compares the performance of the heuristic and brute-force schedulers for operation limits of sixteen and twenty. The heuristic scheduler performs best with more available operations. When limited to sixteen operations, the two schedulers accumulate nearly the

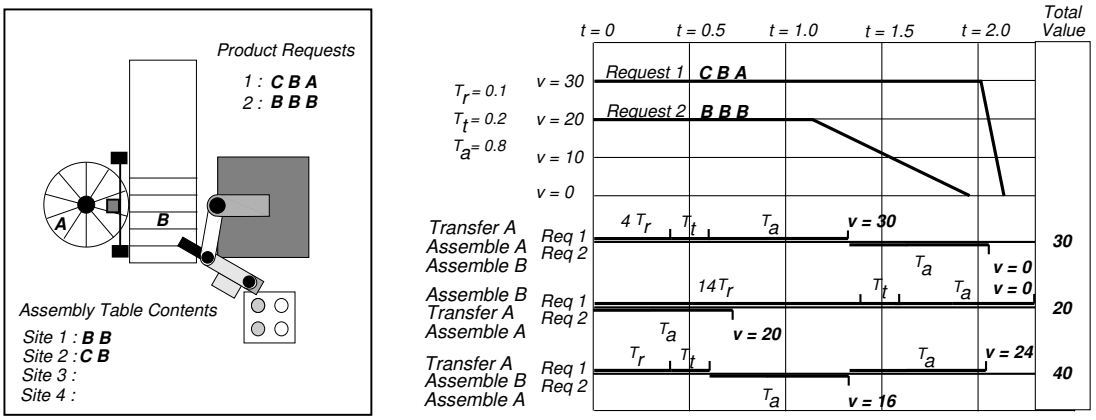

FIGURE 3
Initial state and temporal simulation.

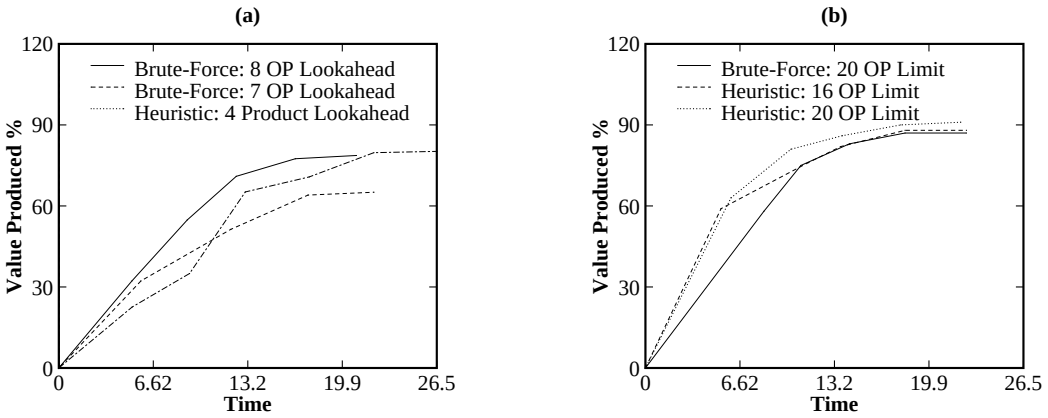

FIGURE 4
Comparative performance of FG schedulers.

same payoff, because the heuristic scheduler cannot fully exploit its four-*product* lookahead advantage over the brute-force scheduler's eight-*operation* lookahead.

III.3. The Anytime Scheduler

The anytime (AT) scheduler uses a resource-bounded reasoning approach called *anytime planning*. An anytime planner allows specification of its execution time and can guarantee an incremental improvement in solution quality with time. This improvement is normally given by metrics based on the degree of certainty about the result's correctness, accuracy, or level of detail.[15, 16]

A performance profile describes the quality of an algorithm output as a function of its computation time. To describe the output quality under different inputs, expected performance profiles are acquired by running the anytime algorithm for several iterations, collecting its performance at each iteration, and then averaging over consecutive runs to obtain a statistically representative profile.

Because anytime algorithms exhibit a time-versus-quality tradeoff, the performance profile and the cost of planning (the value penalty for keeping the system idle while waiting for an operation) are used to compute how much time or how many iterations should be given for executing the anytime algorithm.[17] This decision can be made off- or online after each iteration of the anytime algorithm.

In *passive monitoring* mode, the amount of time, or in our case the number of iterations, that the anytime algorithm requires is decided offline, after analyzing at each point whether it is profitable to keep on planning. In *active monitoring* mode, after each iteration of the anytime algorithm a metareasoning procedure decides if it is profitable to spend more time on planning. The decision is based on the following boolean expression:

$$continue = (plan\ \ value(t) < (utility(t+1) - unit\ \ cost)) \tag{3}$$

where *continue* is a boolean that determines if the anytime algorithm should continue executing, *plan value*(t) is the value of the current best plan, *utility*$(t+1)$ is the expected utility of executing one more iteration of the anytime algorithm (the performance profile provides this information), and the *unit cost* is the cost of one extra iteration of the algorithm.

The AT scheduler is described in Figure 5. It is composed of an *anytime planner*, which generates a plan associating parts on the arrival and transfer buffers to assembly sites, and a *dispatcher*, which controls execution of the operations.

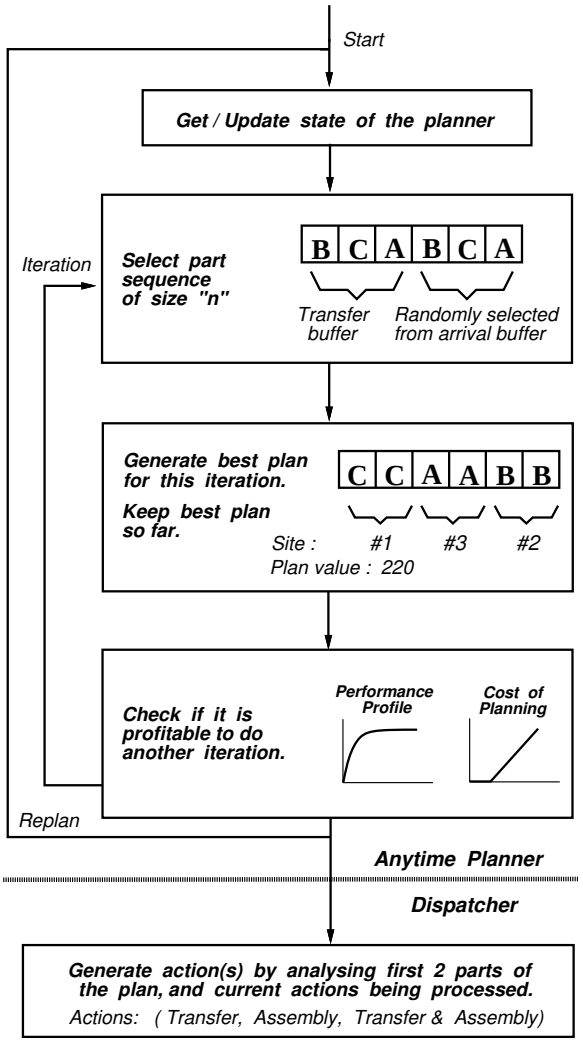

FIGURE 5
Anytime planner and dispatcher algorithms.

The principal design consideration of an anytime planner are:

- *Selection of the right granularity of the problem.* In this step, the size of the initial problem is reduced to speed up computing the planning cycle. In the AT scheduler, n parts out of all possible parts in both arrival and transfer buffers are selected. Constraints involved in the selection of this size n are (1) the time a single plan step should take and the average number of replanning steps necessary to find the plan whose quality satisfies the performance metric, and (2) which intermediate planning algorithm is being used in this step.
- *Selection of the intermediate planner algorithm.* If the task permits, considering the constraints that this decision will impose on the problem reduction described above, an optimal intermediate planner should be selected.
- *The decision as to whether it is profitable to execute another iteration of the planner.*

Considering current requests for products, their value, the parts available in the system, and worst-case execution times of transfer and assembly operations, an exhaustive planner that computes the best sequence of n tuples (parts, assembly sites) was implemented as the intermediate planner of our anytime planner. The AT scheduler replans after every operation, assimilating new arrivals and new or modified product requests.

An example of the technique used by the AT scheduler is demonstrated in Figure 6. In the workstation shown, there are three requests for products in process, each missing two parts. The Gantt chart reveals the plans generated for two selected part sequences of size $n = 6$, $BCABCA$ and $BCABBC$, the former generating the optimal action plan *aabbcc* with total value of 220. The worst-case execution times used are T_{ar} of 2.5 time units for transferring and assembling a part currently at the assembly buffer, and T_{at} of 1.0 time unit for a part currently at the transfer buffer.

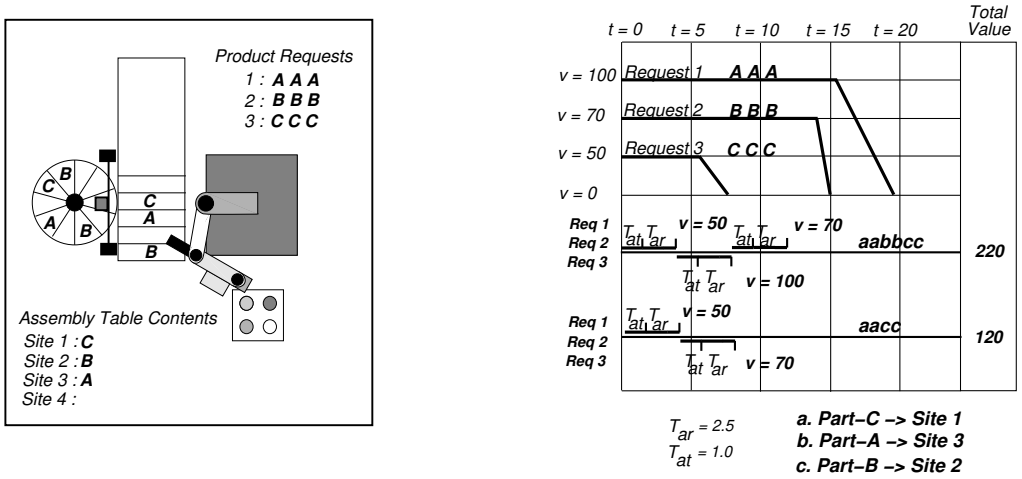

FIGURE 6
Initial state and temporal simulation of two six-part sequence selections.

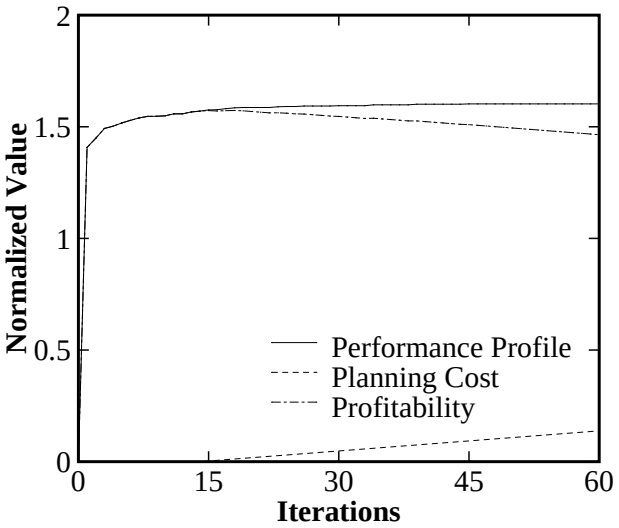

FIGURE 7
Profitability analysis for passive monitoring.

The performance profile of the AT scheduler, a planning cost curve, and the resultant profitability curve (the difference between the performance profile and planning cost curves) are shown in Figure 7. The maximum profitability point indicates the number of iterations that the anytime planner should execute in this case. The anytime planner described in this paper uses active monitoring. (The amount of time an action takes to execute has high variance, and thus the planning cost curve of our system is dynamic, preventing the use of the passive monitoring approach.)

III.3.a. Performance of the AT scheduler. Two versions of the AT scheduler were implemented using different sizes for part sequence. The version that uses part sequence of size $n = 14$ (AT-14) is exhaustive, computing the optimal solution at the granularity level considered in one iteration. However, it cannot operate in real time because it takes too long to generate a plan. Therefore, this version was run offline as a benchmark only. The AT scheduler version under analysis uses part sequence of size $n = 6$ (AT-6). This size selection was based on the real-time constraints of the system and its performance compared to the AT-14 scheduler.

The AT schedulers were tested under different system loads. Figure 8 shows that AT-6 can still deliver a nearly optimal performance, even under unfavorable situations, with buffers near full capacity and product requests with tight deadlines. The curves represent the accumulated value produced over time.

III.4. Workstation Performance

To evaluate the performance of the schedulers under the conditions imposed by the hierarchical control model presented, experiments were performed using a work period of 75 time units. At the beginning of the experiment, all guaranteed

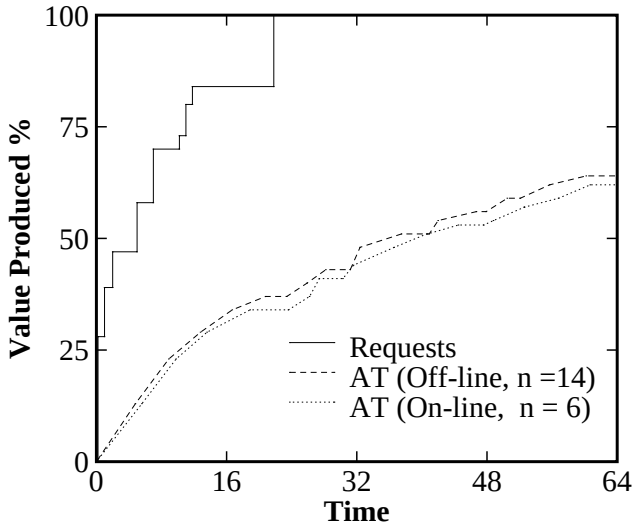

FIGURE 8
Performance analysis of the AT scheduler.

requests for one period (day plan) were loaded from the master production plan, and throughout the run new requests from the next work period were added whenever enough time was reclaimed to guarantee the request. Because only the scheduling level is tested at a single workstation, each request was equipped with a soft deadline and a devaluation rate, but no dynamic priority changes were performed at the production flow level. Figure 9 shows the production and the effects of time reclaiming for both schedulers in one example run, in which the initial value of the guaranteed day plan represents 100%.

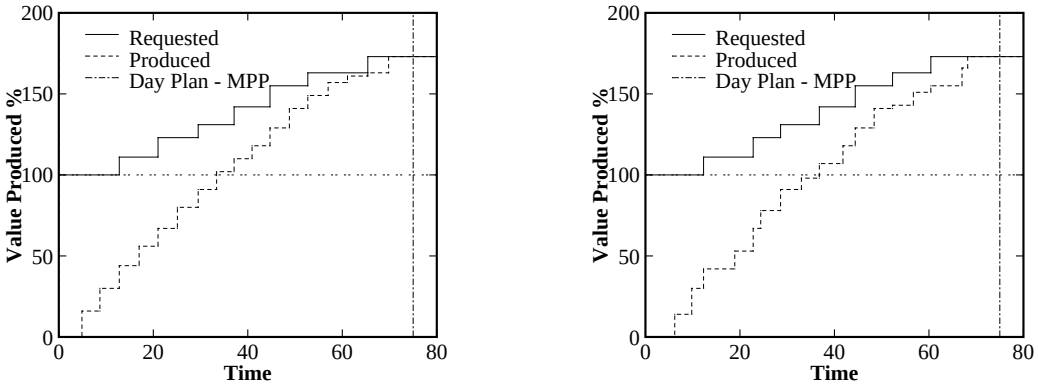

FIGURE 9
Performance of the AT Planner (left) vs Heuristic FG Planner (right).

These graphs show the increases in request value resulting from the arrival of new requests whenever sufficient time is reclaimed, as well as the accumulated value produced throughout the work period. In this example, both schedulers produced an additional 60% without missing any deadlines. Although the final production of both schedulers is nearly identical in these examples, different strength and weaknesses in the approaches can still be found. The FG scheduler's higher timing precision allows it to use the time for buffer alignments more effectively, and it finishes about one time unit earlier than the anytime planner. The anytime approach, on the other hand, promises better scalability to new, potentially more complex systems and part combinations. Whereas the run-time of the FG scheduler's classical planning methods increases exponentially with the size of the problem, potentially leading to long idle times, the anytime algorithm can trade off plan quality for planning time to obtain better overall performance. This suggests that a combination of both approaches in a hierarchical scheduler, with an anytime algorithm planning at the part level followed by a fine-granularity timing analysis of the first part of the resulting plan, would be more efficient.

To show the advantages of value-based scheduling, the performance of the anytime scheduler was compared to a simple best-request-first heuristic. Figure 10 shows the values produced in both cases for one example run.

This example shows that whereas the scheduling schemes described above produce full value, the greedy heuristic violates multiple soft deadlines, resulting in a lower accumulated value for the same requests. The disadvantages of the longer planning times of the value-based scheduling approaches are outweighed by their longer and more valuable schedules, obtained by taking into account the soft deadlines, changed priorities, and available parts. This leads to a higher accumulated value and better long-term utilization of the workcell.

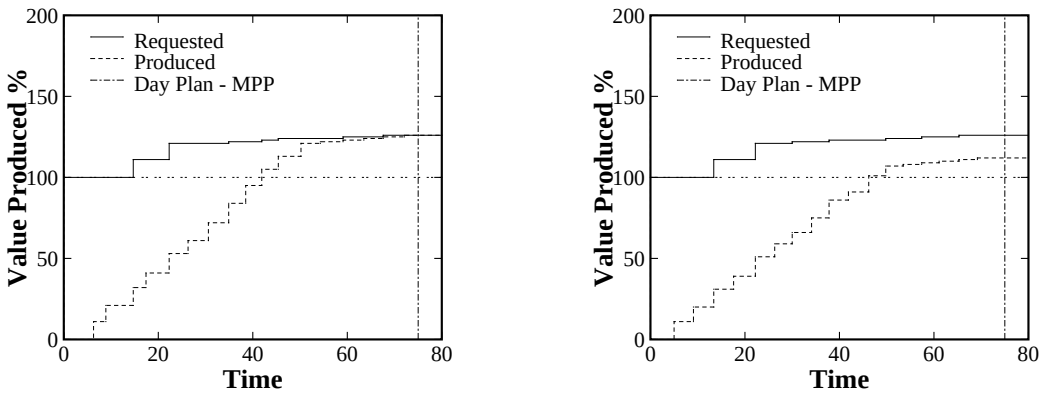

FIGURE 10
Performance of the AT Planner (left) vs Best-request-first Scheduling (right).

IV. CONCLUSIONS

This paper presents a hierarchical model for the control of a flexible manufacturing system; this model consists of a master production level, a flow control level, and a scheduling level distributed among the workstations. Such a scheme promises to make the individual planning and scheduling problems less complex and control of a large number of workstations manageable, without loss of guarantees to the customer.

The main focus of this paper is on the scheduling level, where each workstation opportunistically schedules actions to optimize the accumulated value, given the locally available parts. Uncertainties in part arrival and operation times are handled by reclaiming time gained over the conservative estimates used in the higher levels. Two different schedulers are presented and evaluated: a heuristic fine-granularity scheduler that uses hierarchical planning at two different levels of granularity, and an anytime planner. Although both of these schedulers effectively reclaim time and optimize value and machine utilization, different weaknesses in each approach can be found, suggesting that combining the two methods will yield a scheduler that optimally balances planning time with schedule quality while maintaining a high level of timing precision.

ACKNOWLEDGMENT

This work was supported in part by NSF CDA-8922572 (CII) and IRI-9208920 (CARTS).

REFERENCES

(1) **Agnetis, A., Arbib, C., and Lucertini, M.**, Part routing in flexible assembly systems, *IEEE Transactions on Robotics and Automation*, 6(6):697–705, December 1990.

(2) **Yin, B. and Henrioud, J.-M.**, Adaptive line balancing for flexible assembly systems, In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume III, pages 1980–1985, San Diego, CA, May 1994.

(3) **Stecke, K. and Solberg, J.**, The optimality of unbalanced workloads and machine group sizes in closed queuing networks of multiserver queues, *Operations Research*, 33(4):882–910, 1985.

(4) **Akella, R., Choong, Y., and Gershwin, S. B.**, Performance of hierarchical production scheduling policy, *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, CHMT-7(3):224–240, September 1984.

(5) **Gershwin, S., Hildebrant, R., Suri, R., and Mitter, S.**, A control perspective on recent trends in manufacturing systems, *IEEE Control Systems Magazine*, pages 3–14, April 1986.

(6) **Maimon, O. Z.**, Real-time operational control of flexible manufacturing systems, *Journal of Manufacturing Systems*, 6(2):125–136, 1987.

(7) **Arzi, Y. and Roll, Y.**, Real-time production control of a FMS in a produce-to-order environment, *International Journal of Production Research*, 31(9):2195–2214, September 1993.

(8) **Guthrie, M. A., Lewis, H. J., and Ahmad, M. M.**, Finite scheduling in a make-to-order company, In *Proceedings of the Third International FAIM Conference*, pages 891–902, University of Limerick, Ireland, June 1993.

(9) **Watanabe, T., Kido, K., and Fujii, R.**, The automatic improvement of job-shop schedules and new dispatching rules, In *Proceedings of the 1990 Japan - U.S.A. Symposium on Flexible Automation*, volume III, pages 1133–1138, Kyoto, Japan, July 1990.

(10) **Ho, Y., Eyler, M., and Chien, T.**, A new approach to determine parameter sensitivities of transfer lines, *Management Science*, 29(6):700–714, 1983.

(11) **Rodammer, F. A. and White Jr., K. P.**, A recent survey of production scheduling, *IEEE Transactions on Systems, Man, and Cybernetics*, 18(6):841–851, November/December 1988.

(12) **French, S.**, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Wiley, New York, 1982.

(13) **Burns, L. and Daganzo, C.**, Assembly line job sequencing principles, *International Journal of Production Research*, 25(1):71–99, 1987.

(14) **Sriskandarajah, C., Ladet, P., and Germain, R.**, Scheduling methods for a manufacturing system, In Kusiak, A., editor, *Flexible Manufacturing Systems: Methods and Studies*, pages 173–189. Elsevier, New York, 1986.

(15) **Zilberstein, S.**, *Operational Rationality Through Compilation of Anytime Algorithms*, PhD thesis, University of California at Berkeley, Berkeley, CA, 1993.

(16) **Dean, T. and Boddy, M.**, An analysis of time-dependent planning, In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 49–54, Minneapolis, MN, 1988.

(17) **Horvitz, E. J.**, Reasoning about beliefs and actions under computational resources constraints, In Kanal, L., Levitt, T., and Lemmer, J. F., editors, *Uncertainty in Artificial Intelligence* 3, pages 301–324. North-Holland, 1989.