Fifth Int. FAIM Conference, Stuttgart, Germany, June 1995, pp. 778-789.

Hierarchical Scheduling of Robotic Assembly Operations in a Flexible Manufacturing System *

E. G. Araujo, G. A. Dakin, M. Huber, R. A. Grupen
Laboratory for Perceptual Robotics, Department of Computer Science
University of Massachusetts, Amherst MA 01003 USA

ABSTRACT: An approach to hierarchical scheduling for assembly in a fully-automated FMS 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, planning and monitoring the flow of subassemblies among workstations, and a *scheduler level*, which is concerned with the sequencing and dispatching of 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 mainly on the scheduler level, where heuristics and combinatorial search are combined to achieve near-optimal exploitation of each workstation's capacity. Two scheduling algorithms are described — a *fine-granularity* scheduler, which analyzes robot-conveyor interactions with a fine degree of timing precision, and an *anytime* scheduler, which stochastically generates and evaluates alternative schedules until the tradeoff between expected plan improvements and delays in operation dispatching is no longer favorable.

INTRODUCTION

The majority of manufactured goods in the U.S. are produced in small batch manufacturing processes. This production context places increasing pressure on manufacturing facilities to reduce time-to-market, and to improve response to consumer preference. At the same time, advances in technology allow for an increasing amount of automation even under these conditions.

Flexible, low quantity automated manufacturing poses different challenges than traditional high volume production. While mass production allows the dedication of a whole assembly line to a single product, small batches require rapid job switching to reduce idle time and thus production costs. Such a flexible manufacturing system (FMS) consists of a group of machines, each capable of performing a variety of operations, integrated by an automatic transport facility for transferring parts. Automatic tool changing mechanisms and built-in sensing capabilities

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

enable flexible automation to switch between different operations and thus build a whole suite of products and subassemblies, where many of the final products contain identical materials and subassemblies. The resulting resource sharing requires versatile manufacturing facilities to better cope with uncertainties in part feeding and the different requirements of each product. This flexibility and the required use of sensory input, however, leads to larger variances in task execution times and nondeterministic arrival times for parts and subassemblies. In order to efficiently control such a system, the planning and scheduling level has to be able to accommodate these uncertainties, as well as the resource constraints arising from the coordinated operation of a large number of 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 FMS production management involved in the realization of this plan spans then several levels of detail, from decisions on the production flow level, concerned with production rates and parts routing, to the machine level, which is related to job dispatching and material transport. Much of the research in FMS scheduling has focused on *part routing*, which is concerned with machine selection and route splits within the FMS; and *job sequencing*, which involves the dispatching of parts to each machine in a desirable order.

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], queueing theory [3], and control theory [4 - 6]. While the integer programming formulation yields optimal results, its solution is intractable 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 accommodate occasional disruptions, such as machine failures, which are easier to represent in a control theoretic model, where FMS production flow is modelled 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 applicability 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 efficiently 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 in the case of the routing problem, optimal solutions for the sequencing problem are again 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 employ 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 variances in execution times and nondeterministic part arrivals which severely compromise the quality of *a priori* plans of operation sequences. While statistical treatment of these times might lead to acceptable solutions, idle times can not 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 which attempts to optimize machine utilization in the presence of uncertainties. This decomposition also makes it possible to integrate a large number of workstations. In the next section the overall approach is presented. The remainder of the paper then focuses on scheduling on the workstation level and explores different planning methodologies for operation scheduling. Since the experimental workcell used is equipped with storage buffers, it involves part transferral issues related to linear and circular assembly line scheduling [13, 14], as well as assembly and internal resource sharing.

THE HIERARCHICAL PRODUCTION MODEL

In order to cope with the computational complexity of a large FMS as well as uncertain operational timing, the approach presented in this paper employs a hierarchical production scheduling model. Here different parts of the planning are distributed among different levels and individual workstations. The production management system is shown in Figure 1 and is comprised of three levels: the *master production*, the *production flow*, and the *scheduler level.*

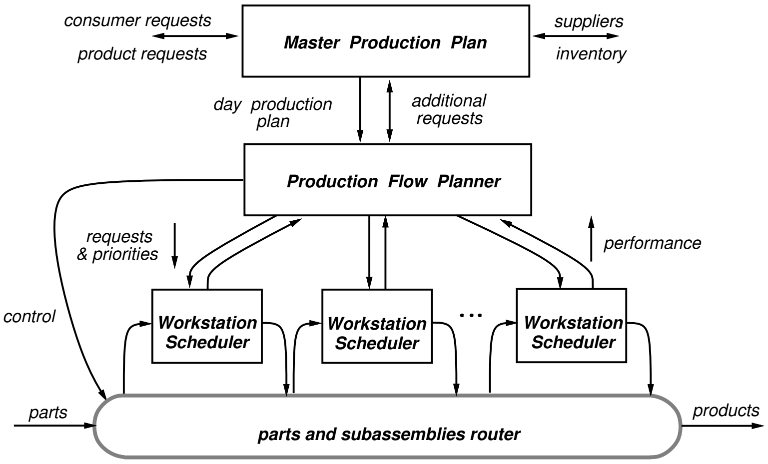

Figure 1: Hierarchical Production Model

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

At the beginning of each work period, the production flow planner distributes the subassembly requests necessary for the products of the period's master production plan evenly among the request buffers of suitable workstations. At the same time it also sends out material request to the storage area. Throughout the production process, this level has three main coordinating tasks based on performance feedback from the lower scheduling level: it controls the part routing, performs monitoring and adjusting of flow rates, and assigns additional 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, but all decisions are based on conservative capacity estimates, performance feedback from the individual workstations, and the state of the routing system.

The scheduler level, finally, 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, produced subassemblies, and the amount of time reclaimed with respect to the worst case production times used for the guarantees in the master production plan.

To account for uncertainties and the opportunistic character of the operation scheduling at the workstation level, all levels have to be able to continuously update their plans to accommodate the reclaimed time and performance feedback. The rate at which feedback is passed on and has to be incorporated at each level, however, decreases drastically with each step upwards in the hierarchy. This allows the whole system to give long-term guarantees while maintaining responsiveness to timing uncertainties.

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, in order to optimize the accumulated value, given the content of its request buffer and the materials currently available to the workstation. The value V of each subassembly, therefore directly influences the production flow of the whole system. This allows the production flow level to coordinate the system by selecting initial values and modifying them on-line in order 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 a *base priority*, P_j, to it, based on the importance or price of the corresponding product. In order to represent temporal urgency more explicitly, this priority is augmented with an artificial, "soft deadline" t_j, to yield a time-varying *product utility* $U_j(t)$:

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

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

Since most of the workstations produce subassemblies that will serve as the 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. In order to achieve this, the production flow level derives 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 the actual and desired flow rates of this subassembly type. Thus, for example, V_j increases when production of the subassembly type k lags behind, resulting in a compensatory increase in 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 factor in the production flow of the system is the routing of materials and subassemblies to the different workstations. This function is also performed by the production flow planner. Whenever parts enter the system or new subassemblies are produced, the flow control level decides to which workstation to route them 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 currently has the least amount of schedulable operation time is selected in order to avoid idle times in the future. The part is then routed there and kept in a local buffer, accessible directly by the workstation.

Both mechanisms, part routing and the modification of scheduling values, 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 and thus allows for larger numbers of workcells in the system. It also allows each workcell to opportunistically schedule its own requests in order 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: a *fine-granularity* (FG) scheduler and an *anytime* (AT) scheduler.

VALUE-BASED SCHEDULING IN A ROBOTIC ASSEMBLY WORKSTATION

Scheduling in each workcell is of critical importance 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 the nondeterministic production and part arrival times, *a priori* schedules would be inefficient, requiring on-line scheduling, based on the currently available parts and subassembly values.

The planning of optimal action schedules is currently an active research area in AI. The combinatorial character of this problem often renders exhaustive planning infeasible since 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 to accept suboptimal solutions. Changing the granularity of the plan is one means to change the complexity of the problem. While it increases when planning on fine grain actions, complexity can be curbed by using coarser schedules on the part or even product level. This coarse granularity, however, might lead to a need for more than one level of planning since high level primitives can not be executed directly. Another way to avoid long idle times is the use of *anytime algorithms.* These algorithms allow the specification of their execution time, and therefore represent a way to overcome time constraints by optimally trading off computation time versus plan quality, based on performance expectations.

In the following, these techniques were incorporated into different value-based schedulers and evaluated on an example workstation.

The Assembly Workstation

In order 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.

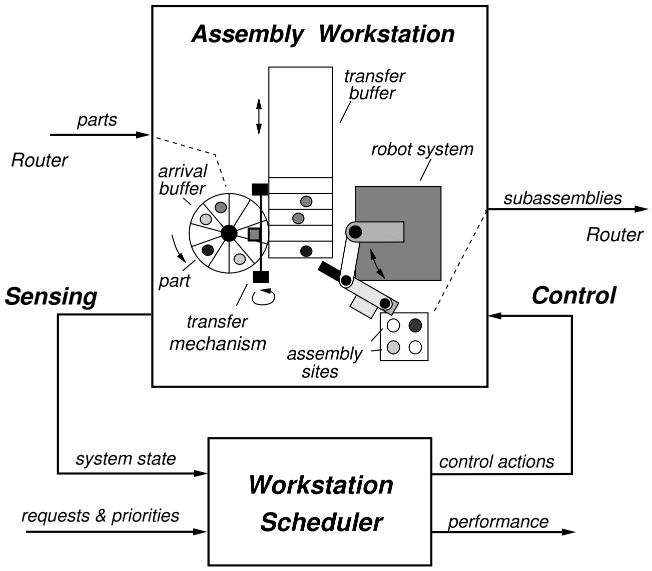

Figure 2: FMS workstation model.

This assembly workstation 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 the *arrival buffer* and to automatically transfer parts. Viewed by a camera, these parts are visually localized on the *transfer buffer*, allowing the robot to acquire and transport them to an assembly site, where assembly operations are performed. The presence of the *arrival buffer* with potentially many parts, and of multiple assembly sites allows the system to plan which requests to build, and to interleave them in order to react to new requests and changes in value. The role of the scheduler is then to coordinate buffer movements, part transferals, visual sensing, and assembly operations such as to maximize the accumulated value of the produced subassemblies, according to the priority scheme described in the previous section. The schedule has thereby to obey given precedence and mutual exclusion constraints between the operations. In the following sections, subassemblies (products) consists of 3 parts, requiring a total of 6 operations for completion.

The Fine-Granularity Scheduler

The FG scheduler analyzes operation sequences through the "simulation" of buffer motions and robot-buffer interactions. Operation sequences are compared in the manner of 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 – a "brute-force" version and a heuristic version. The brute-force FG scheduler performs an unconstrained search among alternative operation orderings, optimizing the combined product values of completed products. Because the search for an optimal ordering takes exponential time in the number of operations, the brute-force scheduler cannot tractably plan many operations into the future, and it is discussed here only for comparison with the heuristic scheduler.

The heuristic FG scheduler contains two hierarchical planning levels: high-level product selection and low-level sequencing of operations. At the top level, the scheduler selects a sequence of products to assemble, heuristically estimating the time of each product's completion, and the associated product value. The temporal estimate is obtained by assuming constant operation times, ignoring context dependencies at run-time. This coarse but efficient temporal analysis permits the heuristic scheduler to plan with a lookahead window of several products. Lower-level planning then proceeds as in the brute-force FG scheduler, but only the operation sequences required to complete the first product in the chosen product sequence are enumerated. Both FG scheduler versions replan their operation sequences after every *assemble* operation, to incorporate any 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

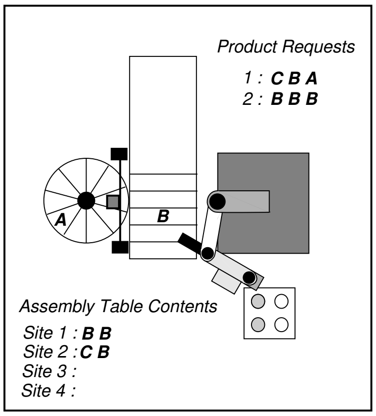

Figure 3: Initial state.

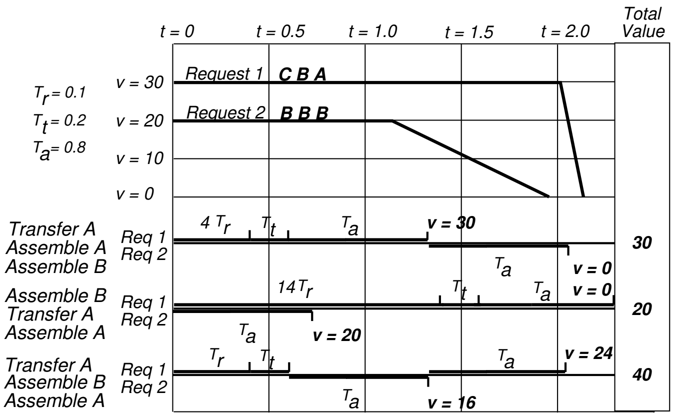

Figure 4: Temporal simulation.

rotates at a rate of $T_r = 0.1$ time units per bin; part transferals to the transfer buffer require $T_t = 0.2$; and robotic transport from the transfer buffer to the assembly table, plus the assembly thereafter, 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 3 operations may now be performed: *transfer A*, *assemble A*, and *assemble B*. Out of feasible 3 operation sequences, the planner must decide which sequence optimizes the combined values of the two completed products. The Gantt chart in Figure 4 reveals that the third operation sequence, which interleaves the two products' operations, yields the greatest sum (40) of product values. Although this sequence satisfies neither soft deadline, it optimally balances the products' relative priorities.

Performance of the FG scheduler. While the heuristic FG scheduler requires comparatively little run-time, and always finishes within the time-span 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 "off-line", running the scheduler and workstation simulator non-concurrently.

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 5 show the incrementally increasing sums of product values achieved by the different FG scheduler versions. Figure 5(a) contrasts the behavior of the "greedy" brute-force scheduler with the heuristic scheduler's greater lookahead. The brute-force scheduler initially achieves greater payoffs than the heuristic scheduler, whose 4-product lookahead leads to superior long-term performance. The brute-force scheduler similarly achieves greater payoffs with a lookahead window of 8 operations than with 7. Brute-force scheduling experiments were not performed for lookahead windows larger than 8 operations, due to CPU time constraints. Figure 5(b) shows scheduler performance for two different 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. Figure 5(b) compares the performance of the heuristic and brute-force schedulers for operation limits of 16 and 20. The heuristic scheduler performs best when there is a greater number of available operations. When limited to 16 operations, the two schedulers accumulate nearly the same payoff, since the heuristic scheduler cannot fully exploit its 4-product lookahead advantage over the brute-force scheduler's 8-operation lookahead.

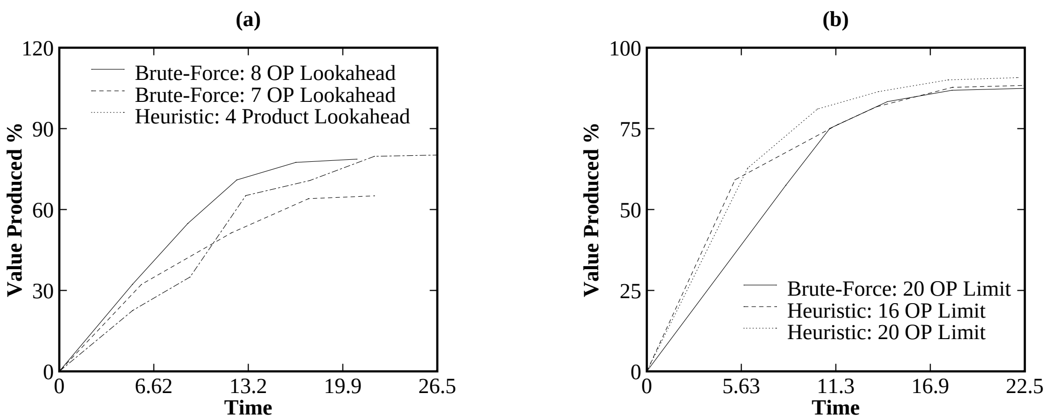

Figure 5: Comparative performance of FG schedulers.

The Anytime Scheduler

The AT Scheduler was implemented using a resource-bounded reasoning approach called *anytime planning*. An *anytime planner* allows the specification of its execution time, and can guarantee an incremental improvement in solution quality with time. This performance improvement is normally given by metrics based on the degree of certainty about the correctness of the result, degree of accuracy of the result, or the level of detail of the result [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.

Since anytime algorithms exhibit a time versus quality tradeoff, its *performance profile* and a function that describes the *cost of planning*, value penalty for keeping the system idle waiting for an operation, are used to compute how much time or how many iterations should be given for the anytime algorithm execution [17]. This decision can be made off-line or on-line, 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 executes, is decided off-line after analyzing at each point whether or not it is profitable to keep on planning. In *active monitoring* mode, after each iteration of the anytime algorithm, a meta-reasoning procedure decides if it is profitable to spend more time on planning. The decision is based in 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 6. It is composed of: an *anytime planner* that generates a plan associating parts on the arrival and transfer buffers to assembly sites, and a *dispatcher* that controls the execution of the operations.

The principal design consideration of an anytime planner are:

- Selection of the right granularity of the problem. In this step, the initial problem is reduced to a size that allows a speed-up in the computation of 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 the time a single plan step should take and the number of replanning steps necessary on average to find the plan whose quality satisfies the performance metric used, and 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 planner should be selected to be the intermediate planner.
- The decision if it is profitable to execute another iteration of the planner.

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

An example of the technique used by the AT scheduler is demonstrated in Figure 7. In the workstation shown, there are currently three requests for products in process, missing two parts each. The Gantt chart reveals the plans generated for two selected part sequences of size $n = 6$, $BCABCA$ and $BCABBC$, the former being the one that generated 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

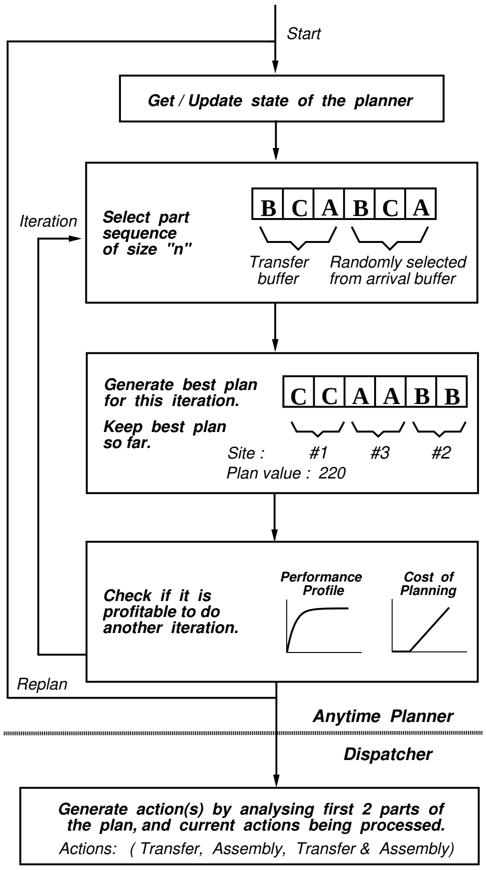

Figure 6: Anytime planner and dispatcher algorithms.

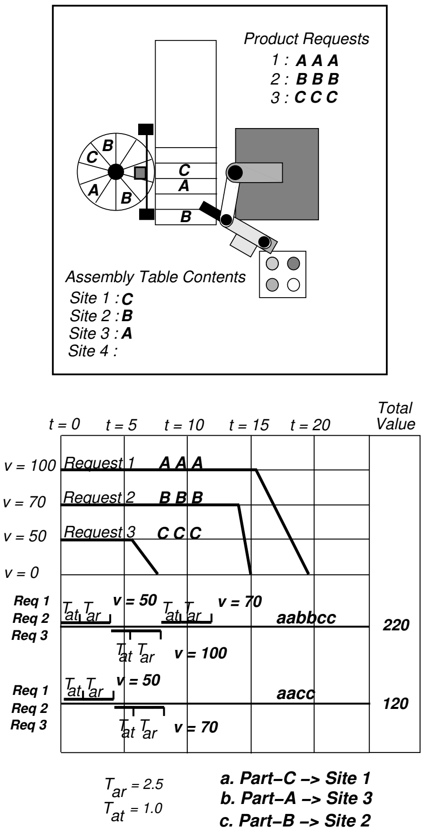

Figure 7: Initial state and temporal simulation of two different six part sequence selection.

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.

The *performance profile* of the AT scheduler, a *planning cost* curve, and the resultant *profitability* curve (difference between the performance profile and planning cost curves) are shown in Figure 8. The maximum profitability point indicates the number of iterations that the anytime planner should execute on 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.

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 can not operate in a real-time situation, since it takes a considerable amount of time to generate a plan. Therefore, this version was only used as a benchmark, running "off-line". The AT scheduler version under analysis uses part sequence of size $n = 6$ (AT-6). This part sequence size was selected 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 9 shows that AT-6 can still deliver a near-optimal performance even under specially unfavorable situations with buffers near full capacity, and product requests with tight deadlines. The curves represent the accumulated value produced over time.

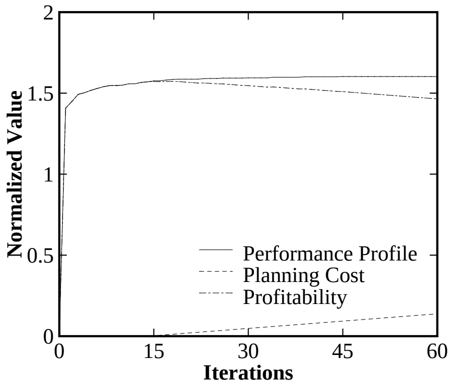

Figure 8: Profitability analysis for passive monitoring.

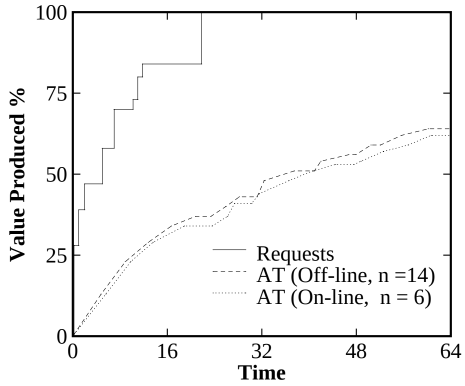

Figure 9: Performance analysis of the AT scheduler.

Workstation Performance

In order to evaluate the performance of the schedulers under the conditions imposed by the hierarchical control model presented in this paper, experiments were performed using a work period of 75 time units. At the beginning of the experiment all guaranteed requests for one period (day plan) are loaded from the master production plan and throughout the run, new requests from the next work period are added whenever enough time was reclaimed to guarantee this request. Since only the scheduling level is tested at a single workstation, each request is equipped with a soft deadline and a devaluation rate, but no dynamic priority changes are performed by the production flow level. Figure 10 shows the production and the effects of time reclaiming for both schedulers in one example run where the initial value of the guaranteed day plan represents 100%.

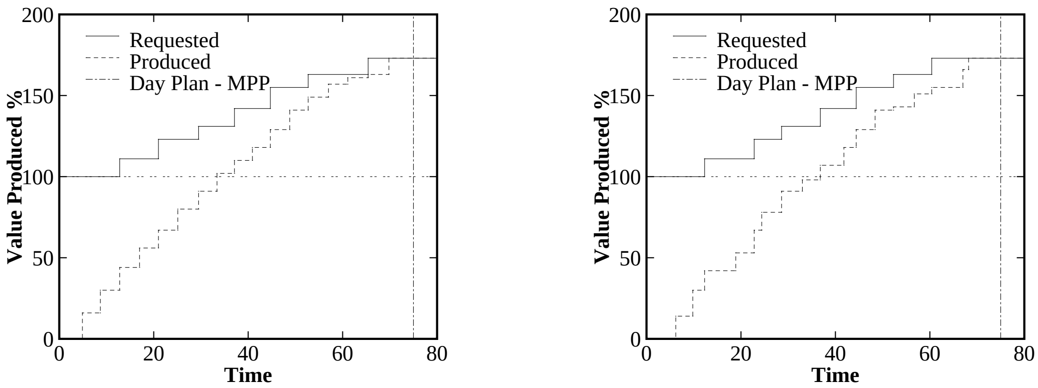

Figure 10: Performance of the AT Planner (left) vs Heuristic FG Planner (right).

These graphs show the increases in request value due to 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 succeed in producing an additional 60% without the missing of any deadlines. Although the final production of both schedulers is nearly identical on these examples, different strength and weaknesses of the approaches can still be found. The higher timing precision of the FG scheduler allows it to use the time for buffer alignments more effectively and thus finishes about 1 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. While 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 the utility of a combination of both approaches in the form of a hierarchical scheduler with an anytime algorithm planning at the part level, followed by a fine grain timing analysis of the first part of the resulting plan.

To show the advantages of value-based scheduling, the performance of the anytime scheduler was compared to a simple best-request-first heuristic. Figure 11 shows the values produced in both cases for one example run.

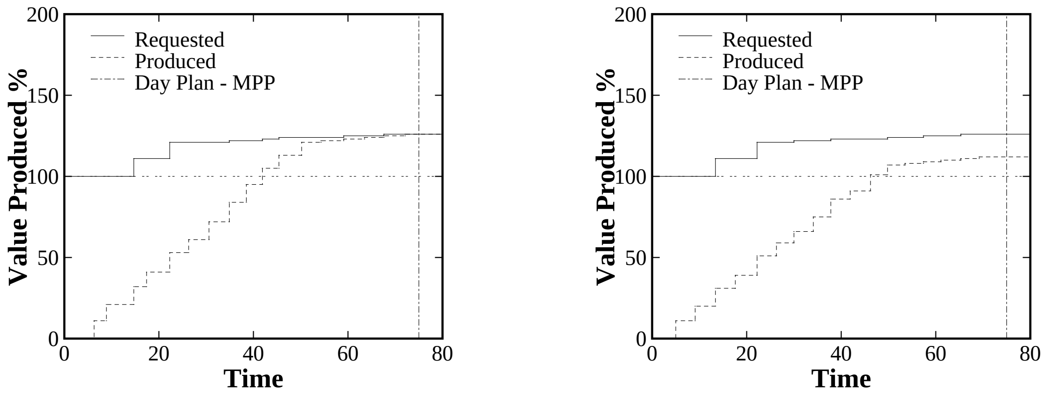

Figure 11: Performance of the AT Planner (left) vs Best-request-first Scheduling (right).

This example shows that while 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 longer planning times of the value based scheduling approaches are here outweighed by the longer and more valuable schedules obtained by taking into account the soft deadlines, changed priorities, and available parts, leading to a higher accumulated value and potentially better long-term utilization of the workcell.

CONCLUSIONS

This paper presents a hierarchical model for the control of a flexible manufacturing system, consisting of a master production, a flow control, and a scheduling level, which is distributed among the workstations. Such a scheme promises a reduction in the complexity of the individual planning and scheduling problems and thus the capability to control FMSs comprised of a large numbers of workstations without the 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. Here uncertainties in part arrival and operation times are handled by reclaiming gained time compared to the conservative estimates used in the higher levels. Two different schedulers, a heuristic FG scheduler which uses hierarchical planning at two different levels of granularity, and an anytime planner are presented and evaluated. Although both of these effectively reclaim time and optimize value and machine utilization, different weaknesses of each approach can be found, suggesting the utility of combining the two methods to yield a scheduler which optimally balances planning time with schedule quality while still maintaining a high level of timing precision.

REFERENCES

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

[2] B. Yin and J.-M. Henrioud. 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. IEEE.

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

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

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

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

[7] Y. Arzi and Y. Roll. 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] M. A. Guthrie, H. J. Lewis, and M. M. Ahmad. 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. M. M. Ahmad and W. G. Sullivan (Ed.).

[9] T. Watanabe, K. Kido, and R. Fujii. 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. ISCIE.

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

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

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

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

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

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

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

[17] E. J. Horvitz. Reasoning about beliefs and actions under computational resources constraints. In *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence*, Seattle, WA, 1987.