Adaptive Scheduling for High-Volume Shops

Carlos F. G. Bispo, João J. S. Sentieiro, *Senior Member, IEEE*, and Roger D. Hibberd

Abstract—**A conceptual algorithm to solve scheduling problems designed to provide accurate reactive scheduling for high-volume shops is presented. Due to the NP-hard nature of these problems, one must resort to heuristic approaches, where dispatch scheduling appears to be one of the most successful. The proposed algorithm was developed to provide classical dispatch scheduling with an embedded optimization procedure. The algorithm delivers successive updates of the initially generated schedule by extending in an unconventional way the horizon of applicability. An adaptive mechanism is implemented. The successive schedule updates are generated using beam search. A particular instance of the conceptual algorithm is presented and discussed; this implementation is motivated by the need to solve high-volume scheduling problems where a periodic production pattern is assumed to be a constraint to satisfy. Analysis of the algorithm's behavior reveals that the performance measure converges to the calculated lower bounds and that the computational time can be polynomial, for each problem, with the number of parts to process.**

I. Introduction

In production systems, the definition of resources, products, the production process for each product, and production volumes is made in the broad context of *planning*. Once these problems have been solved, the main goal is to plan day-to-day production. At this level, the questions under consideration are related to resource allocations to several tasks (necessary to complete the products) and to their sequencing in time so that a predefined criterion or performance measure is optimized. This is the *scheduling problem* [3], [10].

A production system is composed of a set of machines, which are able to execute a predefined set of operations; a set of parts to be processed, Ω; and a set of transport devices that provide the flow of the parts between machines and between these and the central store (warehouse). Typically the parts to process are classified by type, where each type is characterized by a particular sequence of operations (usually denominated technological or precedence constraints) necessary to convert raw material (or only partially processed products) into a final product or a subproduct eventually used to assemble a final product. For the classes of systems under consideration in this paper, this sequence is fixed and known *a priori* for each type, and the operation duration times are assumed to be deterministic. Also, no particular relation among the sequences for the different types is assumed.

The scheduling problem under consideration in this paper is defined in the context of *high-volume shops*. High-volume shops are understood here as models where the total number of parts to be processed is very high, but the total number of different types is low when compared with the total volume of parts (low-variety systems).

Because the number of parts in Ω is very high, the use of search techniques in the generation of a schedule for the complete set is a heavy task due to its NP-hard nature. Moreover, a complete schedule for the jobs is not sufficiently flexible, since unexpected events may jeopardize its utilization, wasting the time spent on its generation. This consideration implicates the need to obtain *schedulers* that not only provide efficient schedules but are also able to deal with the changing environment of production systems, i.e., schedulers that are *accurate*, producing optimal or near-optimal schedules, and *reactive*, able to react in useful time to changes such as machine failures, variations in demand, etc. Therefore, when the objective is to generate schedules for real production problems, it is frequently necessary to resort to heuristic methods where dispatch scheduling strategies assume an important role.

Dispatch scheduling is an appropriate methodology to develop reactive scheduling systems. Based on a single criterion or on a combination of different criteria, dispatch scheduling decides which operation will be executed next. Decisions are taken whenever a resource becomes available. The time spent on schedule generation is usually negligible, since it is based on a relatively simple computation. A great number of different dispatching rules have been proposed in the literature. Examples of the most classical rules are the shortest processing time (SPT) rule, where the choice is made based on the operation with the shortest duration; the earliest due date (EDD) rule, where the choice is made based on the operation corresponding to the part that is due earlier; the most work remaining (MWKR) rule, which selects the operation belonging to the part with the greatest total processing time remaining; and the most operations remaining (MOPNR) rule [10]. In general, although the use of dispatching rules does not ensure optimality of the generated schedules, they usually lead to good solutions in the sense that some balance between optimality and complexity has to be established. The fact that the above-mentioned rules for general scheduling problems are generally short sighted and static in structure has motivated a major research effort toward defining more accurate and dynamic rules for particular optimization criteria [15], [16], [18].

Other techniques have been proposed. Of special interest is the attempt to define heuristic constructive algorithms de-

Manuscript received October 3, 1990; revised February 3, 1992. Portions of this work were presented at the First International Conference on CIM, Troy, NY, 1988.

C. F. G. Bispo and J. J. S. Sentieiro are with the C.A.P.S./Laboratório de Robótica e Processamento de Informação, Complexo I–IST, Av. Rovisco Pais, 1096 Lisboa Codex, Portugal.

R. D. Hibberd is with the Centre for Robotics and Automated Systems, Imperial College of Science, Technology and Medicine, Exhibition Road, London SW7 2BX, United Kingdom.

IEEE Log Number 9203047.

signed to capture, as closely as possible, the optimal solution structure. Recent examples can be found in [1] for the job-shop problem and in [5] for the flow-shop problem; both use the total production time (TPT) as the performance criterion to optimize. These approaches are very efficient in computational terms, but they are optimization criterion dependent and problem structure dependent.

For high-volume shops, some authors reduce the production control problem to control of the release dates of different jobs into the system, assuming a demand-driven production system and trying to keep the work in progress at low levels [2], [11], [14]. This approach is very efficient when dealing with machine failures, but it does not take into account detailed scheduling.

Other authors approach high-volume scheduling problems using *periodic scheduling*. The concept of periodic scheduling was introduced by Hitz for flow-shop models [12], [13]. The idea is to break the set Ω, when feasible, into small subsets of identical composition Ω_m, accurately schedule one of these subsets, and then simply repeat the schedule over and over until all parts are processed. This periodic strategy assumes that it is possible to define a subset of Ω for which the elementary and repeatable schedule is generated.

This divide-and-conquer strategy was used in [9] for the general job-shop problem, where Ω_m is designated as a *minimal parts set*. For example, in a system where the purpose is to produce 1000 parts of type A, 3000 parts of type B, and 2000 of type C, the minimal parts set would be composed of six parts, one of type A, three of type B, and two of type C, or, eventually, a small multiple of these numbers. Schedule generation is search based, and the objective is to minimize the TPT of Ω_m. Ω_m can almost always be defined. This strategy yields a low flow time for the parts, since each one is processed in a time that at most equals the TPT for an Ω_m.

However, minimizing the TPT for an Ω_m does not necessarily result in the best schedule for Ω. In fact, the objective is to minimize the TPT for Ω, but most frequently the final performance can be significantly deviated from the calculated lower bounds for the total production.

A new concept, called *adaptive scheduling*, is proposed in this paper [4]. It implements a rolling horizon procedure using a new horizon-extension philosophy. It provides accurate scheduling by embedding an optimization procedure within the classical dispatch framework. Also, it is reactive in the sense that it can readjust previous schedules to new system status in a very effective way. This method was established to provide the same type of reactiveness as dispatch scheduling, but with a better degree of optimality. It was motivated by high-volume scheduling problems where a periodic production pattern was assumed to be a constraint to satisfy.

Systems where it may be desirable to have periodic patterns in the output are, for example, those with assembly in the downstream or, less obviously, in demand-driven systems with a low variety. In the latter case, assuming a steady demand for each type, the purpose of satisfying each individual demand can be converted into satisfying some sort of periodicity on the production output. For instance, a system that produces three different types of parts (A, B, and C) with the demands being, respectively, d, $2d$, and d, is intuitively equivalent to a system with a periodic pattern such that, in a single period, one type A part, two type B parts, and one type C part are produced, as long as the period is such that the demand is satisfied in the long run.

In the next section, the conceptual algorithm is defined and discussed. A particular instance of the conceptual algorithm is implemented in Section III. The performance analysis of the algorithm is discussed in Section IV. Simulation results show that the computation time of the proposed algorithm can be polynomial with the number of parts to be processed and that the achieved performances converge to the computed lower bounds on the optimal solution for each problem. The paper concludes with some general considerations on the potentialities of the conceptual algorithm in the context of general scheduling problems.

II. Adaptive Scheduling: A Conceptual Algorithm

The approach proposed in this paper, though different from the periodic scheduling idea, is supported on the definition of small sets for which the use of search-based methods is feasible, and brings to scheduling problems a new concept of a *rolling horizon*.

The key idea underlying the proposed methodology consists of generating successive partial schedules[2] as the production evolves, instead of generating a full schedule for all parts (either by using search or a constructive algorithm). The basic strategy of the approach is to compute a complete schedule for a small set of parts (Ω_S — a *rolling group*), but to apply only its first steps, as in traditional rolling horizon procedures. As production progresses, more parts are included in the rolling group according to a predefined *extension criterion*, which constitutes the novelty relative to the traditional rolling horizon procedures, since those procedures work by extending a fixed amount of time at the end of the original schedule.

When the initial set of parts is extended, another complete schedule is generated, replacing the previous one. This process of schedule generation, partial execution, and extension is repeated until all parts belonging to Ω are produced. This is the core of the *adaptive scheduling algorithm*, described here in detail.

Conceptual Algorithm

Step 1 — Initialization: Define the initial set Ω_S, and the extension and the optimization criteria.

Step 2 — Schedule generation: Generate a schedule S for Ω_S using a search-based approach. Consider the schedule S to be a list of ν decisions, $S = \{d_1, d_2, \ldots, d_\nu\}$, where d_j is the jth decision, for $j = 1, 2, \ldots, \nu$. Set $k = 1$.

Step 3 — Schedule execution: Apply to Ω_S the decision d_k. Set $k = k + 1$.

Step 4 — Horizon extension: If due to the previous decision it is necessary to extend Ω_S, then update Ω_S with

[2] By partial schedule, in this context, we mean a complete schedule for a subset of Ω.

the necessary parts, according to the extension criterion and go to step 2. Otherwise, continue.

Step 5— Stopping condition: If $k \leq \nu$ go to step 3. Otherwise stop.

Remarks:

1) The set Ω_S is called the *schedulable parts set* and constitutes the rolling group.
2) ν will not necessarily be the same for the different Ω_S sets.
3) Each schedule is generated for a rolling group of parts. As long as this group is kept relatively small, the search is feasible. On the other hand, search-based schedule generation makes the approach optimization criterion and problem structure independent.
4) The search procedure starts from the present state of production and implements what is called *rescheduling* in [19].
5) For any given partial schedule,[2] the algorithm will be efficiently reactive to strong changes in the production system (such as machine breakdowns, unexpected operation delays, and changes in production requirements) if the time spent in the search process is not too high when compared with system's time constants.
6) The algorithm has the potential to provide increasingly accurate schedules, since not only are they generated using an optimization procedure (search) but also because they are generated for larger horizons than the ones for which they are executed. As production evolves, more information will be taken into account. This is a shared property with the predictive control approach in control theory [6]–[8].
7) The extension criterion has a role similar to a release policy [2], [11], [14].

To implement a particular instance of this algorithm, it is necessary to define the model of the system, the composition of the initial rolling group Ω_S, the optimization criterion, the search control structure, and the extension criterion.

III. Adaptive Scheduling: One Implementation

In this section, we present a particular implementation of the proposed adaptive scheduling algorithm.

A. Working Model

The model of the production system is characterized by the following assumptions:

- Each machine can perform a set of different operations, defined and known *a priori*. There may be operations that can be performed by more than one machine.
- Each machine can only process a single part at a time, and it is assumed that it has a local load/unload buffer of limited capacity (in the model considered in this paper, this limit is one unit).
- Once started, each operation has to be completed without interruption, i.e., there is no preemption.
- The parts in Ω can be of different types. Each type is characterized by a fixed and *a priori* known operation sequence, with deterministic duration times.
- The number of different part types in Ω is much less than the total number of parts, i.e., the parts in Ω exhibit a low variety.

[2] Again, partial schedule means a complete schedule for a subset of Ω.

B. Implementation Details

As previously emphasized, each implementation of the conceptual algorithm requires the definition of

- the composition of the initial Ω_S,
- the optimization criterion,
- the search control structure, and
- the extension criterion.

1) Composition of Ω_S and Optimization Criterion: In the particular implementation considered in this paper, the initial Ω_S is made equal to a minimal set (Ω_m) and the optimization criterion is the TPT for each Ω_S.

2) Search Control Structure: The generation of a schedule plan for all parts in Ω_S is done using a suboptimal search in the *nondelay schedule space* [10] known as *beam search* [17].

Beam search is a suboptimal branch-and-bound search technique. The nodes of the decision tree, representing partial schedules, are organized in different levels indexed by the number of decisions already taken. Each node of a particular level expands into its successors by considering all possible decisions. To evaluate all the successors at a particular level, a dispatching rule is used to complete them. For future expansion only the best β successors are kept, where β is the beam width. Some versions of beam search incorporate a filtering procedure (filtered beam search) where, prior to the above evaluation, a heuristic function is used to select the best α successors, where α is the filter width [17]. Notice that in this later technique each successor will have a lower and an upper bound on the value of the optimal schedule, the first provided by the heuristic function and the second by the completion using the dispatching rule.

To generate and evaluate the alternative schedules, the algorithm uses a discrete event simulation package (scheduler), which is an upgraded version of the one developed in [9]. This upgraded version supports the generation of schedules from any system state instead of doing it only for the initial state of production as in [9], where all machines are idle and no parts in the minimal set have initiated their processing. This feature, allowing what is called rescheduling in [19], is of paramount importance for the adaptive scheduling algorithm's implementation.

According to the characterization of Ow *et al.* [17], this search package does not incorporate a filtering procedure and has a beam width of one. This means that at any decision stage only the best successor of the nodes under evaluation is kept for further search. During search a simple first-come first-serve (FCFS) dispatching rule is used to complete the partial schedules in order to obtain the global upper bound on the optimal solution. The optimization procedure implemented by beam search is thus exploited to improve the classical dispatch, namely, the performance of FCFS.

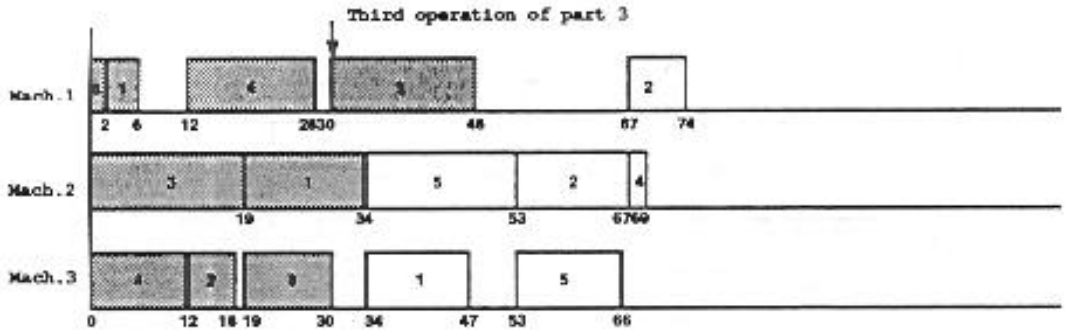


Fig. 1. Schedule for the initial Ω_S.

3) Extension Criterion: The extension criterion defines when new parts have to be included in Ω_S, their number, and their types. It can be activated at any time during production as a function of the system status. This characteristic provides the algorithm with a constant feedback mechanism. The definition of the extension criterion plays a crucial role in the way the algorithm performs. The traditional rolling horizon procedures work by generating plans for *a priori* fixed horizon lengths; after an *a priori* fixed amount of time, a new plan will be generated for the same length. With the proposed extension mechanism the horizon lengths are not fixed, depending only on the problem at hand and on its status, and rescheduling occurs when necessary and not at arbitrary instants of time. Obviously, this implies more frequent rescheduling, but it allows planning for smaller sets of parts.

In the proposed implementation, a set of very simple extension criteria was tested. They differ only in the operation that activates the extension of Ω_S. For the first criterion, the operation that activates the extension is the first one. For the second criterion, it is the second operation, and so on. Thus, for the ith criterion, denoted Op.i, whenever a part starts the ith operation of its sequence, a part of the same type will be included in Ω_S. Due to the periodic production constraint, the newly introduced parts will be assigned lower priorities, implying that, when parts of the same type compete for the same resource, the part with higher priority will be selected first. This will prevent new parts from starting an operation before all the parts of the same type previously included in Ω_S have undergone the same operation, ensuring that the output closely matches the periodic pattern and that the parts keep a low average flow time.

C. A Small Example

To provide a better understanding of the several steps of the algorithm execution, a simple example is presented.

Consider a problem with three machines and Ω_m with five different parts. The machine sequence and the corresponding operation duration times are shown for each type in Table I. The objective is to optimize the TPT, and the extension criterion used is Op.3.

In step 1, the algorithm sets Ω_S equal to the Ω_m defined above, and takes TPT as the optimization criterion and Op.3 as the extension criterion. Step 2 generates the first schedule for Ω_S. The Gantt diagram displayed in Fig. 1 represents the obtained schedule.

TABLE I
MACHINE SEQUENCE AND DURATION TIMES FOR THE FIVE DIFFERENT TYPES

Type	Machine			Duration		
1	1	2	3	4	15	13
2	3	2	1	6	14	7
3	2	3	1	19	11	18
4	3	1	2	12	16	2
5	1	2	3	2	19	13

In step 3, the execution of this plan starts, and the algorithm iterates until the extension criterion is activated, which happens in the ninth decision of the first plan when part 3 (type 3) initiates, on machine one, its third operation.

Step 4 provides the extension of Ω_S by including one part of type 3 (the sixth part) and the algorithm jumps back to step 2, where a second plan for the extended Ω_S is generated. This new schedule is displayed in Fig. 2, where the light grey blocks represent the part of the schedule that will be executed and the dark grey blocks are the part previously decided.

Then, at step 3, the execution of this plan starts at time instant 30. The algorithm then proceeds iterating until rescheduling is necessary, which happens at time instant 34, when parts 1 and 4 simultaneously initiate their third operation. At this point step 4 is executed by adding two new parts to Ω_S, one of the same type as part 1 and the other of the same type as part 4 (parts 7 and 8, respectively). For this new extension another schedule will be generated (step 2) taking into account that machine 1 is busy until time instant 48, machine 2 until 36, and machine 3 until 47.

This process goes on until the whole set Ω is processed.

IV. ALGORITHM ANALYSIS

For the purpose of evaluating the algorithm's performance, a FORTRAN 77 implementation of the adaptive scheduling algorithm was tested on a C220 CONVEX computer (with a processing speed of 30-50 MIPS) for a set of generated problems. A three-stage analysis was carried out.

In the first stage, the algorithm's performance is studied as a function of the extension criterion in terms of the TPT, the parts average flow time (AFT), and the computational

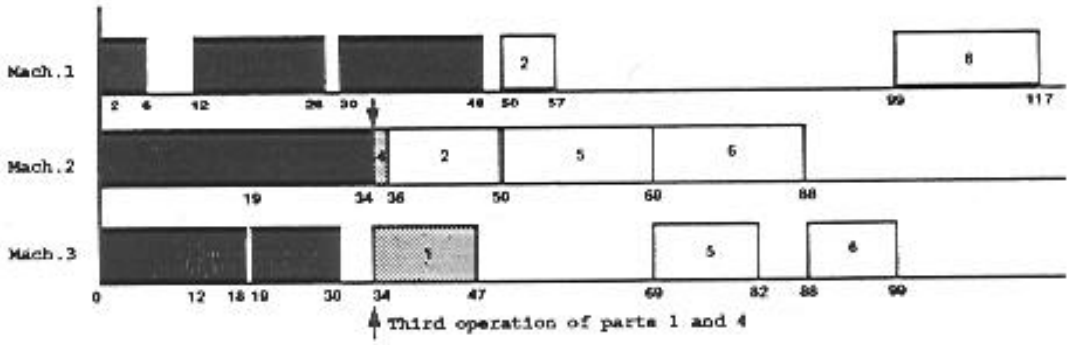


Fig. 2. Schedule for the extended Ω_S.

TABLE II
PROBLEM 1—FIVE MACHINES AND TEN PARTS IN Ω_m

	Op.1			Op.2			Op.3			Op.4			Op.5		
n	CPU	TPT	AFT	CPU	TPT	AFT	CPU	TPT	AFT	CPU	TPT	AFT	CPU	TPT	AFT
10	1.6	776	546.5	1.6	776	546.5	1.6	776	546.5	1.6	776	546.5	1.6	776	546.5
20	25.9	1400	757.6	18.5	1390*	690.1	9.9	1390*	622.0	7.3	1390*	625.0	5.0	1428	576.0
30	68.9	2085*	846.6	31.7	2085*	701.2	23.0	2085*	670.5	12.1	2085*	599.9	8.3	2162	578.7
40	133.0	2780*	964.0	54.1	2780*	854.2	30.8	2809	705.7	18.4	2780*	642.2	11.8	2819	589.4
50	237.3	3475*	1131.9	81.7	3585	938.1	43.1	3475*	732.5	24.7	3475*	666.0	15.8	3475*	608.1
60	357.6	4170*	1269.9	97.3	4170*	969.9	51.6	4170*	759.0	30.9	4255	681.5	20.0	4170*	609.7
70	536.9	4865*	1389.4	134.5	4897	1037.2	66.0	4865*	802.1	37.2	4865*	690.5	22.9	4865*	607.7
80	732.3	5592	1489.4	169.7	5592	1112.6	80.4	5560*	817.9	45.0	5560*	715.4	26.4	5572	609.2
90	977.7	6255*	1616.4	209.0	6255*	1179.0	93.6	6268	856.2	51.7	6291	725.9	30.6	6268	624.3
100	1272.3	6967	1744.9	262.4	6968	1248.9	108.2	6957	871.1	58.7	6950*	733.2	34.2	7074	615.3
200	—	—	—	—	—	—	—	—	—	—	—	—	75.5	14042	624.5
500	—	—	—	—	—	—	—	—	—	—	—	—	191.6	34802	635.2

*Value known to be optimal.

time (CPU). Several problems are tested and results are presented for a representative example. In the second stage, the algorithm's dynamic behavior is studied by observing the evolution along time of several features, namely, the machine utilization rates, the size of the rolling group, the average flow time, the rescheduling horizon, and frequency. In the third stage, a set of different dimension problems is simulated using the *best extension criterion* and the results discussed. The best extension criterion is defined as the one that leads to a polynomial increase of the CPU time with the size of Ω, ensuring at the same time the lowest TPT.

For all the tested cases, the operation sequences are randomly generated and the operation duration times are integers uniformly distributed in the interval [5, p. 99].

A. Influence of the Extension Criterion

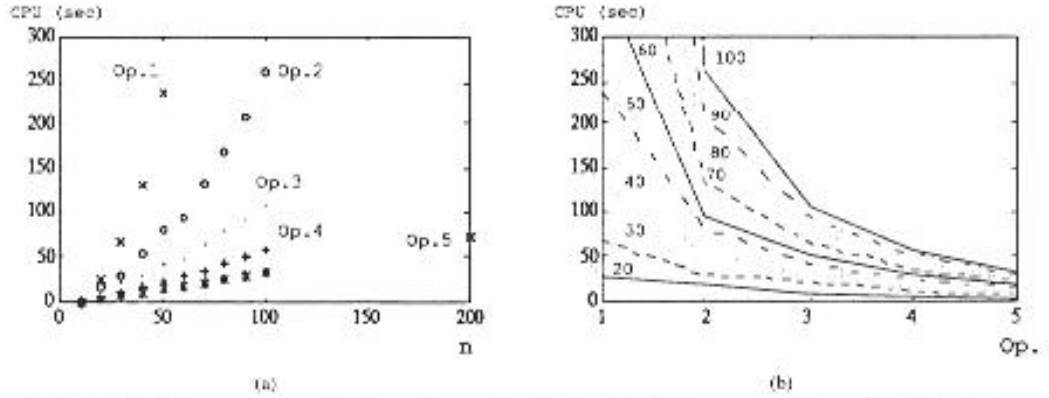
Consider a job-shop problem with five machines and an Ω_m set with ten parts, where all machines are different and each part needs an operation on each machine.

The adaptive scheduling algorithm is applied to this problem, with five different extension criteria. The corresponding results are displayed in Table II where n denotes the number of parts in Ω.

The lower bound (LB)[3] for this problem is 695 for each Ω_m. This means, for example, that for the production of 200 parts LB is equal to 13900. Since the cost achieved by the algorithm is 14042, the deviation is 1%. To simulate the production of 200 parts, the algorithm has to generate 191 schedules: one for the initial Ω_S with 10 parts, followed by one for each extension, until the last part is included in Ω_S. Since the CPU time necessary to simulate the production of 200 parts (using Op.5) is 75.5 s, the average schedule generation time is 0.4 s.

The results in Table II reveal that the TPT does not degrade substantially when the parts are included later in Ω_S. This is not a general property as will be seen in Section IV-C. Also, when the extension criterion changes from Op.1 to Op.5, a substantial reduction in the computation time is observed for $n > 10$. This can be observed in Fig. 3(b), where the graphs describing the computational time evolution, as a function of the extension criterion, are parameterized by the number of parts n in Ω. In addition, Fig. 3(a) shows that the computational time for Op.1 increases exponentially with the number of parts in Ω as opposed to the almost linear increase observed for Op.5. When using Op.5, the AFT tends toward

[3]The lower bound is defined as the total processing time of the machine with the highest workload.



(a) (b)

Fig. 3. (a) CPU time versus parts in Ω for each extension criterion. (b) CPU time versus extension criteria for each n.

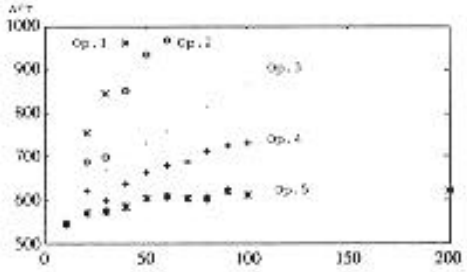


Fig. 4. Average flow time versus number of parts in Ω.

a constant value but is highly increasing for Op.1, as clearly revealed in Fig 4.

For Op.5 and $n > 20$, the TPT exhibits a maximum deviation of less than 4% relative to the lower bound; as n increases, that deviation tends to remain below 1% (in particular, for $n = 500$ the deviation is 0.15%).

The AFT (Fig. 4) increases significantly for the first two criteria and has a moderate increase for the third. For Op.4 and Op.5 the AFT becomes constant as n increases, their values being not too far apart from those achieved for a single set ($n = 10$). In fact, the deviation is 34.2 and 12.6%, respectively, for the fourth and fifth criteria.

For this particular problem, the best extension criterion is Op.5, since it induces the same TPT levels of performance as the others, with a substantial reduction in computational time and a better settling value for the AFT.

The results of this example that can be extendible to other dimension problems are the ones relative to the CPU time and the AFT. In what concerns TPT, the later the parts are included in Ω_S, the poorer are the achieved performances. However, as n increases, the deviation relative to the calculated lower bounds decreases. Therefore, a tradeoff between computational time and achieved TPT will have to be established for each particular problem.

B. Dynamic Evolution

In order to appreciate the dynamic behavior of the solutions provided by the adaptive scheduling algorithm, problem 1 is, for this purpose, considered a representative example. The results refer to simulations obtained with Op.5.

Fig. 5 shows the machine utilization rates of the five machines of problem 1 as a function of time. The rates are steady and correspond to the loads associated with each machine. Machine 2 constitutes the bottleneck and has a production rate close to 100%, confirming near-optimal performance as discussed in the previous section.

This behavior is observed in all simulated cases, where the best extension criterion is used.

The number of parts in the rolling group and the average flow time are strongly connected, i.e., if the rolling group is bounded in size, the AFT will converge to a constant. In fact, the polynomial CPU time was observed only in the cases where Ω_S is bounded, and consequently those are the only situations where AFT becomes bounded as well. Fig. 6 shows the evolution of the Ω_S size and the AFT as functions of the number of extensions for problem 1.

When parts are included earlier, as is the case with Op.1 and Op.2, the size of the rolling group increases without control; the same happens with the AFT. Since search is exponential with the size of Ω_S, this explains the results of Table II.

Other interesting measures of the adaptive scheduling potential for real-time control are the rescheduling frequency and the horizon length. Every time rescheduling is activated, the search package returns a feasible schedule for the present Ω_S, together with the time instant when its production is concluded. In Fig. 7, A describes the evolution of this time instant against the number of reschedulings performed. In the same figure, curve B describes the instants of rescheduling activation. For instance, the hundredth rescheduling is activated at time instant 7037 and the corresponding Ω_S production will be concluded at time instant 7560. The difference, A-B, represents the rescheduling horizon.

Since the horizon length is fairly constant as production evolves, it can be concluded that rescheduling occurs at equally spaced intervals along the production interval. For problem 1, the average interval between consecutive rescheduling is 70.3 units of time (t) and the average horizon length is 663.7.

C. Different Dimension Problems

In order to test the behavior of the adaptive scheduling algorithm relative to different dimension problems, 30 different

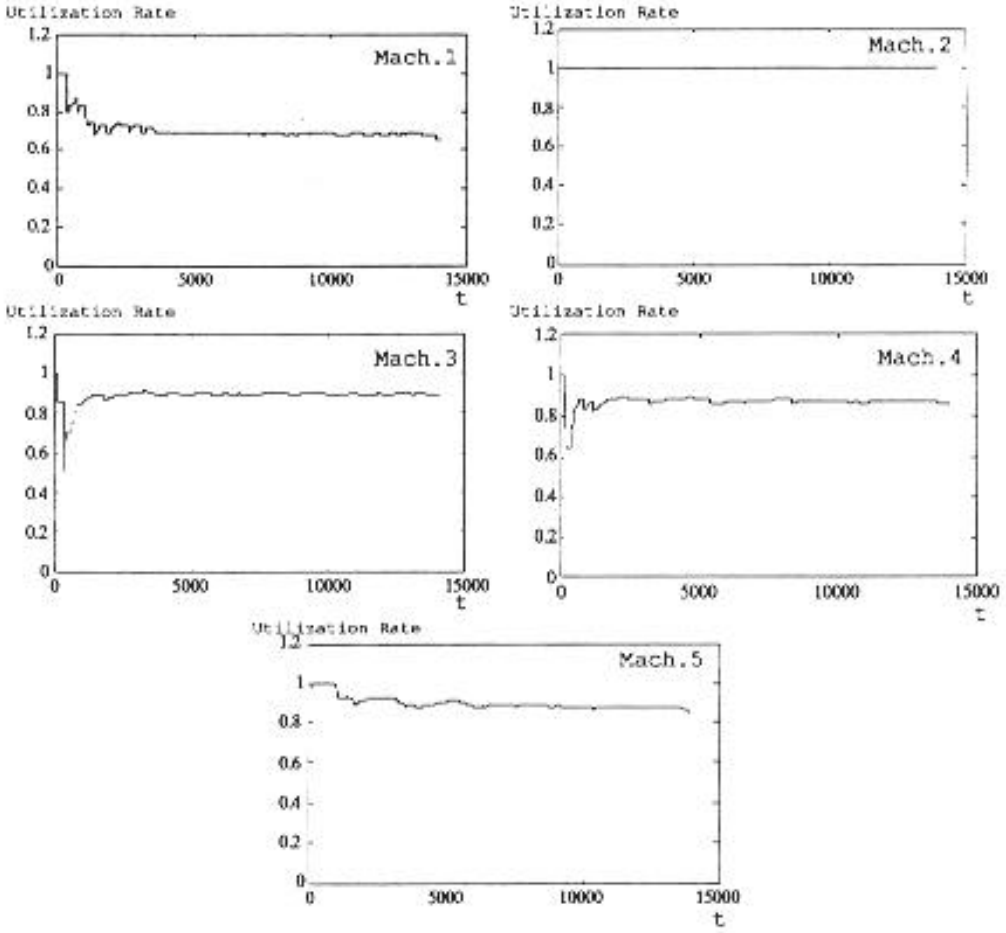
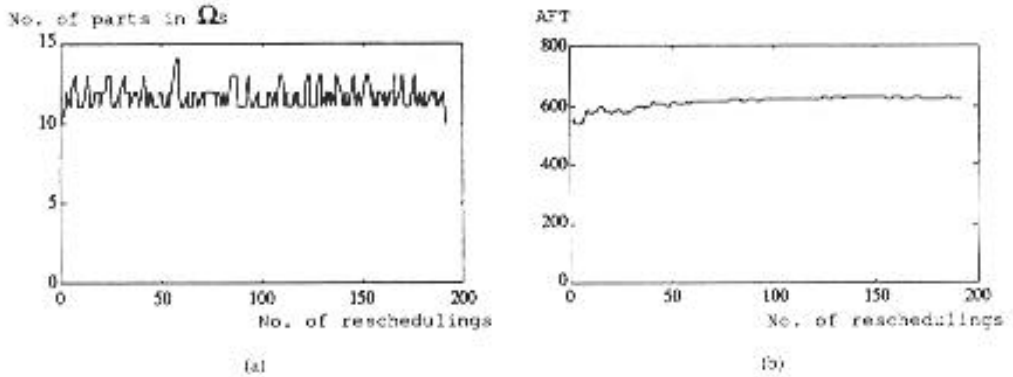


Fig. 5. Machine utilization rates for Problem 1 and a production of 200 parts.



(a) (b)

Fig. 6. (a) Number of parts in Ω_S for each rescheduling. (b) Parts average flow time achieved for each rescheduling.

cases were tested. The results are displayed in Tables III and IV. The extension criterion used is the best criterion for each case.

The results reveal that, as a percentage, the TPT deviation, relative to the LB for each problem, is very small regardless of the initial rolling group size and the number of machines. Cases with a larger number of parts in Ω have also been tested, and the results have revealed that the TPT deviation decreases even more.

The AFT values, for all cases tested, are much smaller than the correspondent TPT values and, as shown in the previous section, do not change significantly with n. This means that the periodic constraint is being satisfied.

The CPU time for each schedule (CPU/sch.) is displayed to emphasize the computational effort involved in each rescheduling when compared with the total production horizon. The polynomial behavior discussed in the previous section is observed again, but the CPU/sch. time for the different dimension

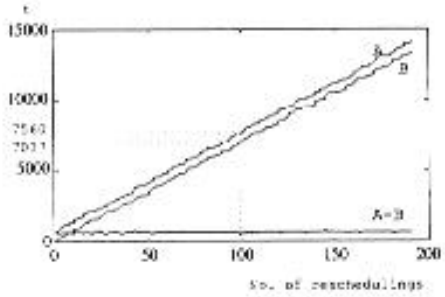


Fig. 7. Rescheduling horizon.

problems grows exponentially with the size of the rolling group and the number of machines. This exponential behavior may be overcome, as will be discussed in the next section.

The polynomial complexity, as a function of the total number of parts to process, is feasible when the size of the rolling group is bounded. In fact, assuming a bound on the rolling group size and a bound on the computational time for such size, the computational complexity of the algorithm is linear with the number of parts in Ω. The extension criterion controls the rolling group size, meaning that appropriate criteria can ensure that the rolling group is bounded.

The fact that the computational time associated with each rescheduling is low makes the adaptive scheduling approach particularly well suited to deal with events like machine breakdowns or changes in production requirements. The partial schedule can be adjusted in a relatively short time to accomodate those events.

D. Computational Aspects

From a computational point of view, the generation of the successive partial schedules could be better exploited if other search packages were used. In fact, the package available was implemented in FORTRAN 77, which is generally recognized as a poor tool to implement search. Also, though beam search is an intrinsically parallel method, and the search algorithm ran in a CONVEX computer, this aspect was not exploited, and the search was done sequentially.

No special effort was made for the improvement of the available package (apart from the upgrading mentioned in Section III-B-2), since this was not the main concern of the research work described in this paper. The use of more efficient search packages would not alter the polynomial behavior (relative to the size of Ω) exhibited by the adaptive scheduling algorithm. One can, however, conjecture that, due to parallelism, a more efficient search could eventually make schedule generation of polynomial type with respect to the size of the rolling group Ω_S.

V. Conclusions

A new approach called adaptive scheduling has been proposed for high-volume production problems with a periodic constraint. The main novelty of the approach is the introduction of a new extended horizon concept.

One implementation was presented and several cases studied. The results were analyzed and some interesting properties confirmed. In particular, the analysis revealed the polynomial behavior of the algorithm when dealing with high-volume problems (with the appropriate extension criterion) and its capacity to deal in a systematic way (without further adaptations) with situations such as changes in production requirements and machine breakdowns.

Note that the polynomial behavior is defined in terms of the number of parts in Ω (not in Ω_S) and is possible as long as the number of parts in Ω_S is bounded. Since the rolling group evolution is controlled by the extension criterion, it is always possible to force Ω_S to be bounded.

A. Adaptive Scheduling versus Other Approaches

When comparing adaptive scheduling with other approaches proposed in the literature, it can be argued, at least in qualitative terms, that adaptive scheduling will eventually perform better than dispatch, release control, heuristic constructive, or periodic scheduling.

Relative to dispatch scheduling, once given a particular rule, the beam search can use that rule (in the nodes evaluation stage) to generate the partial schedules.[4] Therefore, the plans generated by beam search will be, in the worst case, as good as the ones obtained with the given rule, using simple dispatch.

For demand-driven systems where the parts release control is the only concern and detailed scheduling is obtained by dispatch [14], the above reasoning is applicable. Hence, it can be claimed that any dispatch approach can be improved by adaptive scheduling. The release policies can be incorporated as the extension criterion.

Heuristic constructive algorithms usually generate a complete schedule for all parts supported on a particular optimization criterion. Some incorporate rescheduling capabilities or have the potential to exploit them [1], [5]. However, they are designed for a specific optimization criterion and, consequently, are not extendible to other problems. Since in the adaptive scheduling approach the schedules are generated by search, this algorithm can be applyed using other optimization criteria with a simple change of the nodes evaluation function. There is no evidence in the literature that constructive algorithms will perform better as the production volume increases, either in terms of achieved costs or in terms of computational time [1]. The adaptive scheduling algorithm has proved to perform better when the problem dimension increases and can exhibit a polynomial increase in computation time. Relative to the specific optimization criterion under study in this paper, the results presented here are comparable to those shown in [1], provided that the performances achieved in [1] are extendible to high-volume shops.

In general, periodic scheduling does not lead to steady machine utilization rates and only achieves performances similar to adaptive scheduling when the schedule for Ω_m is very close to optimum. In any case, the initial deviation will

[4]This is what constitutes the optimization procedure incorporated in classical dispatch.

TABLE III
PROBLEMS WITH FIVE MACHINES

Problem	n	Ext. Crit.	LB	CPU (seconds)	CPU/sch. (seconds)	TPT	TPT dev. (%)	AFT
5 Machines, Ω_m with 10 Parts								
1	200	Op.5	13900	75.5	0.40	14042	1.0	624.5
2	200	Op.5	12760	85.9	0.45	12830	0.5	563.1
3	200	Op.5	12720	78.7	0.41	12720	0	547.8
4	200	Op.5	12260	74.4	0.39	12360	0.8	536.7
5	200	Op.5	11420	64.4	0.34	11635	1.9	510.4
5 Machines, Ω_m with 15 Parts								
6	300	Op.5	18300	464.2	1.6	18311	0.1	676.9
7	300	Op.5	17900	477.1	1.7	17900	0	726.7
8	300	Op.5	20420	467.1	1.6	20440	0.1	868.3
9	300	Op.5	18680	476.8	1.7	18680	0	739.1
10	300	Op.5	17960	487.0	1.7	17989	0.2	891.7
5 Machines, Ω_m with 20 Parts								
11	200	Op.5	11650	804.6	4.4	11678	0.2	840.5
12	200	Op.5	12080	822.6	4.5	12080	0	1001.7
13	200	Op.5	12320	788.6	4.4	12320	0	880.4
14	200	Op.5	11740	846.3	4.7	11782	0.4	847.9
15	200	Op.5	10940	823.7	4.6	10940	0	776.9

TABLE IV
PROBLEMS WITH TEN MACHINES

Problem	n	Ext. Crit.	LB	CPU (seconds)	CPU/sch. (seconds)	TPT	TPT dev. (%)	AFT
10 Machines, Ω_m with 10 Parts								
16	200	Op.5	12440	720.7	3.8	12753	2.5	1116.9
17	200	Op.5	12240	736.3	3.9	12639	3.3	1174.6
18	200	Op.5	13280	478.7	2.5	13540	2.0	1048.9
19	200	Op.5	13200	553.4	2.9	13530	2.5	1027.9
20	200	Op.5	12140	756.4	4.0	12579	3.6	1091.4
10 Machines, Ω_m with 15 Parts								
21	300	Op.5	19440	1258.0	4.4	19572	0.7	1210.8
22	300	Op.5	19720	1281.7	4.5	20130	2.1	1148.2
23	300	Op.5	19820	1209.5	4.2	20280	2.3	1103.4
24	300	Op.5	19320	1155.0	4.0	19495	0.9	1090.4
25	300	Op.5	19540	1170.2	4.1	19674	0.7	1048.9
10 Machines, Ω_m with 20 Parts								
26	200	Op.10	12130	1435.5	7.9	12355	2.0	1126.8
27	200	Op.10	12990	1629.6	9.0	13100	0.8	1172.0
28	200	Op.10	11850	1367.3	7.6	11906	0.5	1101.7
29	200	Op.10	12140	1369.0	7.6	12404	2.2	1158.0
30	200	Op.10	12790	1448.8	8.0	12839	0.4	1157.0

generally propagate, i.e., if the lower bound deviation for Ω_m is 15%, it will remain constant for Ω, no matter how large Ω is. This is not the case with the adaptive scheduling algorithm. The AFT performance of periodic scheduling is also broadly achieved by adaptive scheduling.

B. Future Research

In order to test the algorithm's performance robustness when applied to other classes of problems, a set of simulations was performed where the periodicity constraint was relaxed.

TABLE V
SAMPLE RESULTS WITHOUT THE PERIODICITY CONSTRAINT

Problem n	Ext. Crit.	LB	CPU (seconds)	CPU/sch. (seconds)	TPT	TPT dev. (%)	AFT
5 Machines, Ω_m with 15 Parts							
31	300	Op.4	17401	656.3	2.3	17751	2.0	707.5
32	300	Op.4	16894	637.8	2.2	17190	1.8	709.7
33	300	Op.4	15040	625.6	2.2	15608	3.8	670.9
34	300	Op.4	17914	606.3	2.1	18373	2.6	658.1
35	300	Op.4	18334	638.4	2.2	18369	0.2	887.0
31	450	Op.4	26595	1009.9	2.3	26804	0.8	716.7
32	450	Op.4	26282	950.8	2.2	26535	1.0	711.0
33	450	Op.4	22360	965.5	2.2	23225	3.9	654.1
34	450	Op.4	26732	944.8	2.2	27168	1.6	678.9
35	450	Op.4	27025	975.2	2.2	27061	0.1	880.7

Table V shows the results obtained under the following conditions:

1) The system is composed of five machines and produces 20 different types of parts.
2) The initial Ω_S is composed of only 15 parts, each of a different type.
3) When the extension criterion is activated, the type of the part to be included in Ω_S is randomly selected from one of the 20 types available, using a uniform distribution on the integer interval [1, 20].
4) The optimization criterion is the TPT for Ω_S.
5) Two sets of simulations were considered: one for 300 parts and the other for 450.

These sets, of five problems each, simulate a production system where there is the same demand for each of the 20 types (the choice of type is made using a uniform distribution in the discrete set [1, 20]). As in the periodic case, it is possible to observe a steady behavior on the AFT values, the maintenance of the same values for the computation times, and the convergence tendency to the computed LB (except for problem 33) when the total number of parts to process increases.

Even though these results are not enough to substantiate a claim of extendibility beyond the periodic constraint, they encourage further research in order to evaluate how the proposed methodology will perform for other classes of problems.

Other topics deserving further consideration can be identified. In fact, the division of the total set of parts into small subsets is not a mandatory condition (as is the case for the above examples). In order to apply adaptive scheduling to problems other than the ones considered in this paper, different extension criteria could be tried. For instance, in situations where parts have an attached due date, their lead times could be used to define the release sequence. To decide when more parts should be released, the extension criterion should take into account issues such as capacity constraints and machine utilization rates. For demand-driven systems it is also possible to define an extension criterion based on the demand pattern and on the system's capacity. The criterion could weigh the rolling group size in order to control the computational time evolution.

Relative to the search package used, two main issues deserve to be mentioned. First, one could take advantage of the fact that beam search is an intrinsically parallel search technique, and thus reduce the computational effort of each rescheduling. The second issue has to do with the particular dispatching rule used to obtain the results presented in this paper. FCFS is recognized as a poor rule. The intent of the work described in this paper was to show how adaptive scheduling can perform. Future work in this area should consider dispatching rules that can provide better decisions than FCFS.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Dr. Z. Doulgeri for discussions during the preliminary phase of this work, to the anonymous reviewers for helpful and constructive comments, and to Ms. B. Slavinsky for editing.

REFERENCES

[1] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Sci.*, vol. 34, no. 3, Mar. 1988.
[2] R. Akella, Y. Choong, and S. B. Gershwin, "Performance of hierarchical production scheduling policy," *IEEE Trans. Components, Hybrids, Manufact. Technol.*, vol. CHMT-7, no. 3, Sept. 1984.
[3] K. R. Baker, *Introduction to Scheduling and Sequencing*. New York: Wiley, 1974.
[4] C. Bispo and J. Sentieiro, "An extended horizon scheduling algorithm for the job-shop problem," in *Proc. 1st Int. Conf. CIM* (Rensselaer Polytech. Inst., Troy, NY), May 1988.
[5] L. Carvalho, R. Almeida, C. Bispo, and J. Sentieiro, "An integrated environment for planning and scheduling in flow-shop manufacturing plants," in *Proc. 2nd Int. Conf. CIM* (Rensselaer Polytech. Inst., Troy, NY), May 1990.
[6] D. W. Clarke, P. P. Kanjilal, and C. Mohtadi, "A generalized LQG approach to self-tuning control," *Int. J. Contr.*, vol. 41, pp. 1509–1544, 1987.
[7] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized predictive control, parts 1 and 2," *Automatica*, vol. 23, no. 2, pp. 137–160, Mar. 1987.
[8] D. W. Clarke and C. Mohtadi, "Properties of generalized predictive control," Intern. Rep., Dept. Eng. Sci., Oxford Univ., Oxford, U.K., Jan. 1988.

[9] Z. Doulgeri, "Production scheduling policy for flexible manufacturing systems," Ph.D. dissertation, Dept. of Mechan. Eng., Imperial College of Science and Technol., London, U.K., June 1987.
[10] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. New York: Wiley, 1982.
[11] S. B. Gershwin, R. Akella, and Y. F. Choong, "Short-term production scheduling of an automated manufacturing facility," *IBM J. Res. Develop.*, vol. 29, no. 4, July 1985.
[12] K. L. Hitz, "Scheduling of flexible flow-shops I," Rep. LIDS-R-879, Lab. for Info. and Decision Syst., MIT, Cambridge, MA, Jan. 1979.
[13] K. L. Hitz, "Scheduling of flexible flow-shops II," Rep. LIDS-R-1049, Lab. for Info. and Decision Syst., MIT, Cambridge, MA, Oct. 1980.
[14] J. G. Kimenia and S. B. Gershwin, "An algorithm for the computer control of production in flexible manufacturing systems," *IIE Trans.*, vol. 15, no. 4, Dec. 1983.
[15] S. R. Lawrence and T. E. Morton, "Resource-constrained multiproject scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics," Intern. Rep., Graduate School of Industrial Admin., CMU, Mar. 1990.
[16] T. E. Morton, S. R. Lawrence, S. Rajagopalan, and S. Kekre, "SCHED-STAR: A price-based shop scheduling module," Intern. Rep., Graduate School of Industrial Admin., CMU, Feb. 1988.
[17] P. S. Ow and T. E. Morton, "Filtered beam search in scheduling," *Int. J. Prod. Res.*, vol. 26, no. 1, 35–62, 1988.
[18] ——, "The single machine early/tardy problem," *Management Sci.*, vol. 35, no. 2, Feb. 1989.
[19] F. A. Rodammer and K. P. White Jr., "A recent survey of production scheduling," *IEEE Trans. Syst. Man Cybern.*, vol. 18, no. 6, Nov./Dec. 1988.

Carlos F. G. Bispo received the M.Sc. degree in electrical and computer engineering from the Technical University of Lisbon, Lisbon, Portugal, in 1988. He is currently working toward the Ph.D. degree at the Graduate School of Industrial Administration, Carnegie Mellong University, Pittsburgh, PA. Prior to his doctoral studies, he was a Teaching Assistant in control and computer integrated manufacturing at the Instituto Superior Técnico (IST), Lisbon. In 1986, he joined the Centro de Análise e Processamento de Sinais/IST Laboratory for Robotics and Information Processing where he was a Research Assistant. His research interests are in the areas of planning and control of manufacturing systems, job shop scheduling, and robotics.

João J. S. Sentieiro (M'87–SM'88) received the Ph.D. degree in electrical engineering from the University of London, London, England, in 1985.

He is currently Professor of Control and Robotics at the Instituto Superior Técnico (IST), Lisbon, Portugal. He is head of the IST Laboratory for Robotics and Information Processing and is a co-founder of the Portuguese Institute for Systems and Robotics. In 1991, he was a Visiting Professor at the Department of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, PA. His research interests include computer vision, robotics, and computer-integrated manufacturing.

Dr. Sentieiro served on the Program Committee of the First European Control Conference, Grenoble, France, and the Second and Third International Conferences on Computer Integrated Manufacturing, Troy, NY.

Roger D. Hibberd received the B.Sc. degree in mechanical engineering in 1969 from Bristol University, Bristol, England, and the Ph.D. degree from the University of London, London, England, in 1977.

He is currently a Senior Lecturer in Mechanical Engineering at the Imperial College of Science, Technology and Medicine, London. He is head of the Manufacturing Technology Group and of the Centre for Robotics and Automated Systems. His research interests include scheduling and control of manufacturing systems, robotic assembly, and the application of robotic technology in surgery and healthcare.