International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:2, No:4, 2008

An Agent-Based Scheduling Framework for Flexible Manufacturing Systems

Iman Badr

***Abstract*—**The concept of flexible manufacturing is highly appealing in gaining a competitive edge in the market by quickly adapting to the changing customer needs. Scheduling jobs on flexible manufacturing systems (FMSs) is a challenging task of managing the available flexibility on the shop floor to react to the dynamics of the environment in real-time. In this paper, an agent-oriented scheduling framework that can be integrated with a real or a simulated FMS is proposed. This framework works in stochastic environments with a dynamic model of job arrival. It supports a hierarchical cooperative scheduling that builds on the available flexibility of the shop floor. Testing the framework on a model of a real FMS showed the capability of the proposed approach to overcome the drawbacks of the conventional approaches and maintain a near optimal solution despite the dynamics of the operational environment.

***Keywords*—**Autonomous agents, Flexible manufacturing systems (FMS), Manufacturing scheduling, Real-time systems.

I. Introduction

FLEXIBILITY in manufacturing plays an increasingly decisive role in keeping pace with the market change world wide. Flexible manufacturing systems (FMSs) are designed to face uncertainties and change in the market by investing on high technology factories that can be controlled to produce wide varieties of products with the same resources. This investment pays back if the flow of production is controlled in a flexible way that adapts in real-time to the changes in job orders and the operation conditions. This implies the demand on flexible allocation of jobs to machines under consideration of the current status of operation as well as the temporal constraints.

The flexibility of the FMSs is enabled by the technological advancement of computerized numerically controlled (CNC) machines. CNC machines offer flexibility on the machine level by possessing the capability of performing different operations. The shop floor of a typical FMS consists of two or more CNC machines connected with an automated transportation system and controlled by a central computer. Each CNC machine is typically equipped with an internal buffer in which cutting tools can be pooled for later use. Carrying out an operation requires setting up the machine with the required tools and downloading a program for controlling the operation [1].

While planning for production, the limited cutting tools are distributed among the machines, which results in partitioning the machines into groups. Each machine group consists of a number of identically or similarly tooled machines capable of performing the same operations. This partitioning is planned according to the expected types and quantities of parts to be manufactured. Parts are likewise grouped into part families based on similarities in physical dimensions and/or processing [2]. This grouping principle adds a new dimension to the gained flexibility by having candidate machines capable of substituting each other in case of failure or overload.

Having alternatives in assigning a part to a machine results in increasing the complexity of scheduling which deals with the real-time management of the flow of production. It involves specifying for each operation of each job a machine to be executed on and a point in time to start the execution at. The big dilemma encountered by scheduling for FMSs is to optimize the performance by utilizing the available flexibility while maintaining real-time reactivity to the dynamics of the operation. A solution that caters for optimality by a thorough investigation of the available alternatives always fails to exhibit real-time reactivity due to the high complexity of the problem.

Conventional methods of scheduling fail to provide a mechanism for reacting to the dynamics of the operation in a timely and effective manner. This operational inflexibility is compensated in practice by skilled personnel who monitor the control status and intervene to adjust the production flow in reaction to disturbances. In addition, attempting to adapt to market changes through the introduction of a new product line or the addition of a new machine is hindered by the high complexity of extending the employed scheduling software [3].

Autonomous agents offer a suitable paradigm for realizing systems dealing with uncertainty and dynamics in a flexible way. Based on their special characteristics like reactivity and interactivity, agents possess a good potential for overcoming the drawbacks of the traditional scheduling approaches.

In this paper, an agent-based framework for manufacturing scheduling is proposed. This framework is based on a hierarchical multi-layer architecture that is abstracted from the manufacturing environment. Scheduling incoming jobs is carried out within the scope of the concerned agents to limit the computational complexity. The generated schedule is

Manuscript received April 29, 2008
Iman Badr is with the Institute of Industrial Automation and Software Engineering, Universität Stuttgart, Germany (e-mail: iman.badr@ias.uni-stuttgart.de).

optimized at different levels of abstractions reflecting the grouping principle employed in FMSs. The proposed framework was tested on a model of a real FMS [4]. Test results prove the capability of the framework to cope with the dynamics of the manufacturing environment flexibly and efficiently.

The paper is organized as follows. Section II analyzes the problem and concludes by identifying the requirements on the desired solution. Section III presents an overview of the agent-based scheduling framework by focusing on the solution approach and architecture, the scheduling method and the advantages of the adoption of agents. Section V discusses an application scenario and test results. Section VI concludes with a summary and an outlook.

II. Scheduling of Flexible Manufacturing Systems

Decisions and tasks concerning the management of production are usually categorized into three interrelated functions: planning, scheduling and control. These are captured in Fig. 1 from a hierarchical view with the corresponding levels of the automation pyramid. Planning is basically concerned with decisions related to the part types, the quantities to be produced and the machine groupings. The plan generated during this phase acts as an input to scheduling and contains data related to the part types to be produced and the corresponding release dates and deadlines.

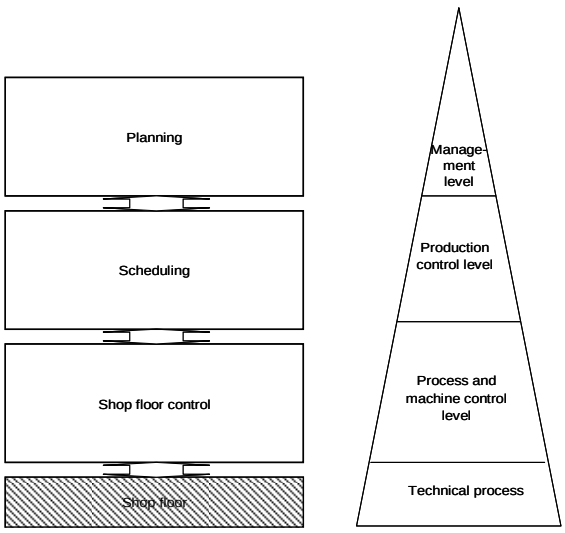


Fig. 1 A hierarchical view of manufacturing control

Scheduling is responsible for finding an allocation sequence for executing the given plan by assigning jobs to resources in real time. The execution of the schedule is performed exogenous to the scheduling by the control layer. However, operating conditions and disturbances like tool or machine failure need to be fed back to scheduling to base its decisions on updated information about the shop floor [5].

A. Problem Definition

Planning results in a set of jobs representing parts to be produced in certain quantities with specific deadlines. For each part type, a set of predetermined operations has to be performed. The task of scheduling is to decide for each operation where and when to be executed. In other words, it involves solving two problems: selecting for each job a specific machine for performing a certain operation and determining the start and end time for the allocated job on the selected machine. While the former is referred to as allocation, the latter is called sequencing. [6].

Scheduling can be conceived as an optimization problem seeking a schedule that minimizes a parameter determined by planning like the average tardiness. It has to take into account the set of constraints related to the parts to be produced such as the technological ordering of operations and the delivery deadline as well as the constraints stemming from the shop floor such as the earliest availability time of machines [7]. Even without considering the dynamics of the environment, this problem is shown to belong to the class of NP complete problems, which makes the search for an optimal solution infeasible due to the high computational complexity. [8].

FMSs are distinguished by a highly dynamic environment, where several events occur all the time that cause deviations from the planned schedule. Changes in planned products as well as disturbances and uncertainties in the operating environment are very likely. Changes in planned products include job cancellation, delay or advance as well as rush orders that may cause changing priorities of planned jobs. Examples of disturbances on the shop floor are over- or underestimation of the execution time, tool or machine failure, and operator absenteeism [9].

B. Conventional Approaches

An attempt to examine the conventional approaches in solving the scheduling problem should differentiate between theory and practice. While researchers have made a remarkably tremendous effort in analyzing the problem and proposing possible solutions, this effort has had limited impact on practice. The problem has always been considered by researches in isolation from the dynamic environment, where conventional algorithms focus on reducing the computational complexity of the problem while ignoring the stochastic and dynamic nature of the environment. This tendency to deal with the problem in a deterministic and static environment has resulted in a wide gap between theory and practice [8].

In practice, FMS scheduling is dealt with by adopting either the centralized static approach or the decentralized dynamic approach [8]. In both cases, dynamics of the environment disturbing the schedule are either left unhandled, or are handled by repairing the schedule according to the expected deviation and its impact.. Repairing a schedule occurs manually by the operators who take corrective actions at the control level like delaying the remaining jobs in case of a disturbance like a machine failure. A regeneration of a new

schedule is only possible in the case of the centralized approach but due to its high computational complexity is only applied under critical situations like a relatively long-term failure of a machine [9], [10].

1) Centralized Static Scheduling

In this approach, a global schedule is generated for the entire system over a certain period of time. This generation usually requires human expertise supported by computerized tools and guided by an online data acquisition system that provides a report on the current status of the shop floor. New jobs that arrive after the schedule generation have to wait for the beginning of the new cycle to be considered.

The advantage of this approach is that all resources are considered while generating a schedule, which can better optimize the utilization of the available capacity of the shop floor. However, due to the computational complexity of generating a global schedule, rescheduling cannot be automated and corrective actions are rather carried out manually as previously explained. Consequently, the availability and productivity of the factory are badly affected.

2) Decentralized Dynamic Scheduling

Another approach which is broadly adopted in practice is the delegation of scheduling to machine control, where one of the parts waiting on the input buffer of a machine is selected based on priority rules. Priority or dispatching rules are rules used to assign priorities for jobs waiting for execution on a certain resource [11]. Assigned priorities are calculated according to parameters related to jobs like arrival time, duration of the operation, or due date.

A large number of priority rules have been proposed in literature. Due to the computational simplicity of most of the dispatching rules, this method is widely adopted. Online scheduling based on dispatching rules or dynamic scheduling is more suitable for the dynamic nature of the job arrival of the FMSs. Rush orders can take higher priority in execution without the manual intervention of operators as in the previous approach.

On the other hand, generating local schedules at the machine level lowers the performance on the global scale. This is due to the lack of consideration of the subsequent operations required for this job and the current status of resources offering these operations. In addition, like the other method, this method fails to automatically readjust to disturbances on the shop floor and the manual intervention in this case is also required. Furthermore, simulation results show that the performance of the various dispatching rules varies according to the operating conditions. Consequently, the application of dispatching rules faces the challenge of the dynamic selection of suitable rules in reaction to the dynamics of the environment.

C. Requirements on Flexible Scheduling

This brief outlook on the conventional approaches reveals the need for another scheduling approach that suits the dynamic nature of flexible manufacturing systems. A number of requirements that the new approach has to fulfill are listed below.

1) Optimizing Utilization of Resources

A good solution should make the best use of the available resources to optimize the scheduling criteria and to maximize the productivity. FMSs are distinguished by a relatively small number of machines ranging from 2 to 30 and a limited number of cutting tools [3]. This dictates the requirement on good utilization of the available resources. Such utilization has take the grouping principle into consideration and manage the collective capabilities of resources grouped together.

2) Adaptability to Changes and Disturbances

The existence of resources with redundant capabilities allows scheduling to automate the reaction to disturbances and dynamics. In this way, the manual intervention is reduced and the real-time performance of the system can be greatly enhanced.

3) Scalability and Extensibility

Conventional scheduling systems are always designed as customized solutions. Changes to the structure of the underlying shop floor or to the planned products by, for instance, introducing a new product is only possible with highly complex and costly major modifications. Consequently, extensibility and scalability should be catered for by scheduling to allow for long-term flexibility [3], [12].

III. A Proposed Agent-Based Scheduling Framework

In this section, the proposed framework is illustrated by focusing on the concepts and highlighting the gained advantages compared to the conventional approaches. The adoption of autonomous agents for realizing the presented concepts is motivated and illustrated.

A. Solution Approach and Architecture

To satisfy the aforementioned requirements, a hierarchical negotiation-based scheduling approach is proposed. In this solution, the problem is decomposed into entities representing machines and jobs. Scheduling occurs dynamically for each job through negotiations between jobs and the required machines. The computational complexity of the problem is hence reduced by limiting the scope of decision making to the concerned entities.

To enhance the performance at the global level within the real-time constraints, optimization is carried out under consideration of the collective capabilities of the entities belonging to the same group. This group-level optimization motivated the introduction of a coordinator entity for every group to manage the group-level capabilities and enforce the real-time constraints. In other words, all jobs related to the same part family are grouped together and coordinated by an entity representing this part family. Similarly, all machines capable of performing the same operation are grouped together under the coordination of an operation entity.

For supporting reactivity, data related to planning and shop floor control are made accessible to scheduling entities. To facilitate scalability and extensibility, entities are defined to be self-contained by encapsulating relevant data of the

corresponding physical entities like machines and part families. The addition of a new machine or a part family would then just mean the instantiation of the corresponding entity.

This hierarchical negotiation-based approach has the following advantages:

1) A change in planning or in the operating conditions is propagated to the concerned entities without affecting other entities. This implies the reactivity to changes with a relatively low computational complexity.
2) The generated schedule is not optimized from just the side of the machines by considering their own local queues like the decentralized scheduling. Each job entity revises the offers and can in case of conflicts or discrepancies negotiate with the corresponding operation entities for possible revision.
3) Disturbances on the shop floor like a break down in one of the required cutting tools could be handled automatically. This is carried out by contacting the corresponding operation entity to take care of delegating the scheduled jobs on that machine to other working machines.

For realizing these concepts, the architecture depicted in Fig. 2 was developed. Scheduling is represented by two main layers representing planning and shop floor control. Each of these layers is further decomposed into two sub-layers for the individual and group-level optimization respectively. These results into four levels of abstractions for machines, operations, jobs, and coordinators of jobs related to the same part family.

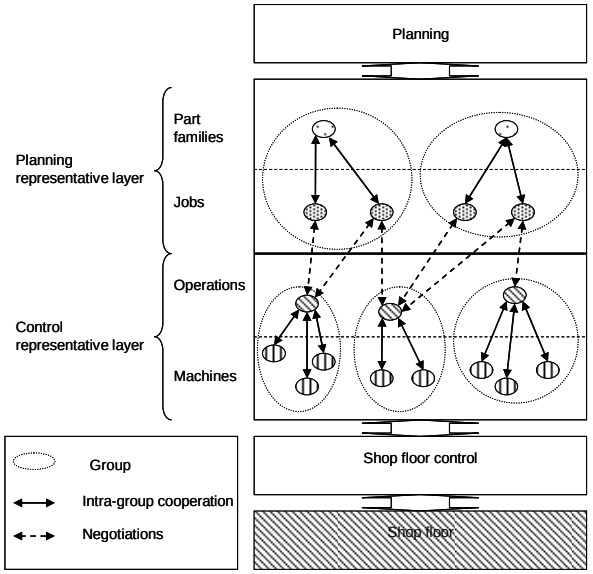


Fig. 2 The architecture of the scheduling framework integrated with other control layers

As illustrated in the figure, scheduling occurs via negotiations between entities of the layers representing planning and shop floor control. This negotiation can be based on the simple contract-net protocol [13]. An illustration of a sample negotiation to generate a schedule for an incoming job order along the four layers of abstraction is captured in Fig. 3.

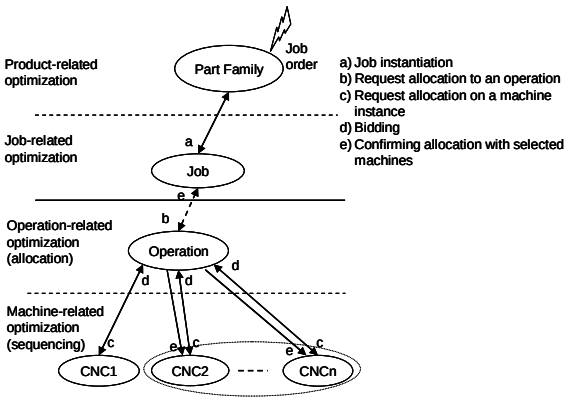


Fig. 3 Illustration of the interaction protocol for the proposed dynamic scheduling

As depicted in the Fig. 3, a job order causes the corresponding product to instantiate a job entity and entitles it to schedule its operations. The job then contacts the required operation entities and requests an allocation under consideration of its temporal parameters like deadline. Each operation forwards the received request to the corresponding machines which reply with their bids that depend on the current scheduled jobs on each machine. The operation selects the best bid(s) and confirms the allocation with the product as well as with the selected machine(s) – surrounded in the figure by a dotted circle.

As illustrated in the figure, the two previously mentioned sub-problems of scheduling are solved at the operation and the machine levels. Where sequencing takes place locally at the machine level, allocation is performed at the operation level to allow for the consideration of the current status of all the member machines and optimize the use of their collective capabilities.

B. Allocation Heuristics

Jobs are characterized by part types and quantities. The sequence of allocating jobs on machines affect the overall performance due to the setup time incurred by the machine in switching from a part type to another. Switching to a part within the same family requires relatively shorter time than switching to another part in another family. While the former is referred to as minor setup time, the latter is denoted as major setup time.

Usually information about incoming jobs is not available a priori. To converge to an optimal solution on the global scale within uncertainty about part types and quantities of incoming jobs, two allocation heuristics are defined. While the first heuristic caters for dealing with uncertainty about incoming part types, the second heuristic attempts to deal with the fluctuations in job volumes. In what follows, the two heuristics are explained.

1) Balancing Resource Distribution (H1)

Major setup time represents a relatively significant overhead. This heuristic attempts to minimize this overhead by balancing the utilization of available machines on the different families of parts. This occurs by allocating a job from a certain part family to a number of machines proportional to the expected quotient of the amount to be produced from this family to the total number of parts along all planned families. Such information can be acquired and updated from planning.

For the sake of minimizing the time lost in setup, an allocation cost is defined to denote the time lost by a machine in switching to a new part. This cost is associated with an allocation request of an incoming part on a certain machine and amounts to the setup time that was incurred by the machine for the previous allocation and would be lost if the machine is to process the new request.

Associating a cost to each allocation bid serves in providing the operation entity with lookahead information about possible effects of the current allocation on the overall performance. Accordingly, the operation entity attempts to minimize the lost setup by selecting the set of machines that offer the bids with the earliest start time and the least allocation cost. Thereby, a reasonably good settlement between the goals of planning and shop floor control is achieved.

2) Greedy Allocation (H2)

Distributing parts to be manufactured among all machines selected by the previous heuristic can be inefficient for jobs with small volumes. This is due to the loss of long setup time for the sake of shorter processing time. To avoid this, the loss represented in the allocation cost is compared to the expected gain represented in the batch processing time. Accordingly, an allocation offer is accepted only if the expected gain exceeds the allocation cost. In the extreme case when the requested allocation deals with a batch size that is too small to yield a gain even on a single machine, the role of the operation entity is to find a way to enforce this allocation.

C. Agent-Based Realization

The realization of the proposed architecture is based on autonomous agents to take advantage of the following characteristics:

1) Reduced Software Engineering Effort

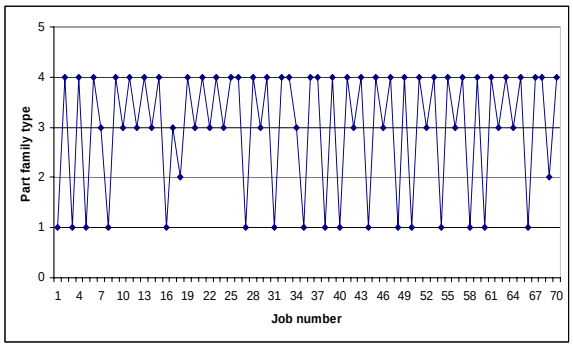
Agent-oriented development facilitates the realization of the aforementioned architecture in two ways. First, the goal-oriented decomposition serves in the separation of the conflicting interests involved in the scheduling problem. Second, the interactions among agents do not need to be statically modelled, which drastically facilitates the development process. Moreover, the agent-oriented abstractions allow the modelling of inter agent communications at a high level of abstraction namely the knowledge level which relieves the developer from the low level details in modelling these interactions [14].

2) Flexibility at Run Time

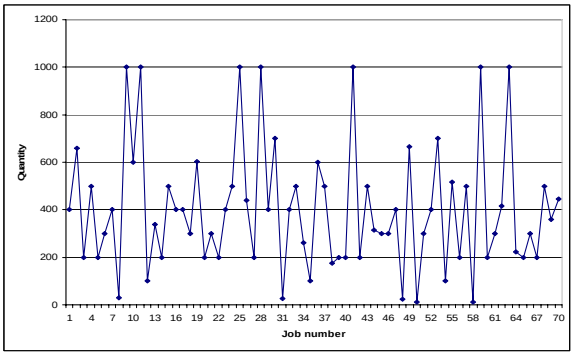
Multi-agent systems are characterized by decentralized autonomous control, where each agent possesses a degree of freedom over its action selection. In this way decisions are made close to the corresponding physical entities which enhances reactivity and adaptability. In addition, multi-agent systems support openness by allowing new agents to register and integrate themselves This feature aids in scalability and extensibility, where the addition of a new machine or the introduction of a new part family would just mean the instantiation of a new agent that can integrate itself easily at run time.

IV. Application and Test Results

The proposed scheduling framework has been tested based on the data of an IBM test line. It represents a huge FMS that consists of thirty one testers grouped into four families. A total of ten different card types grouped into four families are to be tested on one or more testers according to their predetermined plan. The problem is characterized by major and minor setup times. Setup times depend just on the part type being switched to. The goal is to maximize the throughput which is defined as the total number of parts to be manufactured divided by the makespan which is the total time taken to finish all the parts. By assuming a fixed total number of parts over a certain manufacturing period, the problem reduces to minimizing the makespan [4].



(a) Variations of part family types of tested jobs



(b) Variations of quantities of tested jobs

Fig. 4 Illustration of types and quantities of tested jobs

In [4], the reported jobs were based on a static model by assuming that all data already exist before the start of scheduling. Therefore, no information is reported about the details of the individual job arrival over time. The reported data have been hence customized by identifying a set of seventy jobs amounting to a total of 28025 parts based on the reported part types and total quantities. The job arrival pattern was made to feature a great alternation in part types and families between subsequent jobs (see Fig. 4).

The customized dataset was used to compare the performance of the proposed agent-based scheduling approach to the traditional decentralized scheduling approach. The First Come First Serve (FCFS) priority rule was applied for sequencing at the machine level. For the sake of evaluation, the three upper optimization layers of the proposed hierarchy were disabled to emulate the conventional decentralized scheduling based on FCFS. The operation-level optimization was then enabled and tested with the same experimental setting. This was done by testing the performance of applying the first allocation alone followed by testing the two heuristics together.

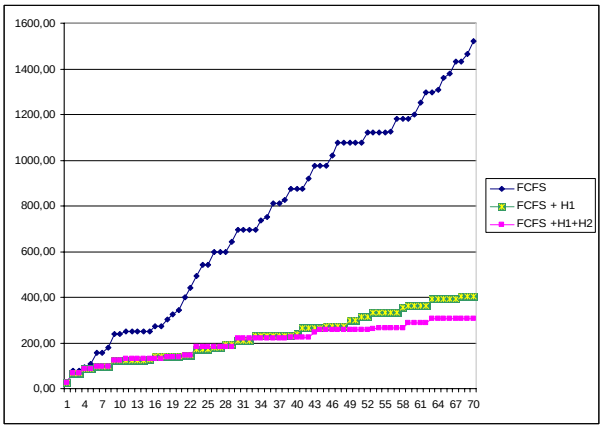


Fig. 5 Effect of variations of job arrival on makespan of the tested scheduling methods

Fig. 5 depicts the variation in makespan in reaction to variations in part family types of subsequent jobs. As illustrated in the figure, the conventional decentralized scheduling led to a dramatic increase in makespan due to the frequent changeover of the setup of the machines which increases the setup time and delays jobs. On the other hand, the cooperative allocation based on the aforementioned heuristics showed flexibility and resistance to change through group-level optimization. Applying the two heuristics together leads to improvement in makespan due to the resulting flexibility in reaction to fluctuation in batch sizes of the planned jobs.

Compared to the lower bound derived mathematically in [4], the performance of the proposed approach lies within 9.9% of the optimal makespan. This is in contrast to the decentralized scheduling whose performance amounted to about 548% of the optimal performance (see Table I). Considering the centralized static scheduling based on a static model of job arrival, where all data about incoming jobs were determined a priori, a schedule within 3% of the optimal solution could be generated. [4]. Due to its lack of support to the dynamic model of job arrival, the centralized approach could not be tested with the modified dataset.

TABLE I
MAKESPAN AND THROUGHPUT OF THE TESTED METHODS RELATIVE TO THE OPTIMAL BOUNDS

Scheduling method	Makespan (relative to lower bound)	Throughput (relative to upper bound)
FCFS	547.9%	18,20%
FCFS+ H1	145.6%	68.6%
FCFS+ H2	109,90%	90,90%

The computational complexity was measured with respect to the, the execution time needed to schedule each job on a 2.0 GH processor. It was found that scheduling one job takes on average around 181,8 milliseconds by applying heuristic1 alone and 161 milliseconds when applying both heuristics. This enhancement resulting from adding the second heuristic can be attributed to the reduction in the number of the machines selected for allocation which in its turn reduces the overhead involved in negotiations among the corresponding agents. Overall, the resulting execution time allows interleaving scheduling with execution of real FMSs.

V. Conclusion and Future Work

In this paper, the architecture and scheduling method of an agent-based scheduling framework of the FMSs is proposed. This framework is based on a hierarchical multi-layer architecture that builds on the grouping principle of FMSs. It optimizes the performance of the generated schedule at several levels of abstractions. The architecture is designed to support scalability and extensibility. Moreover, the flexibility in enabling and disabling optimization at the different levels serves in test purposes.

Test results on a model of a real FMS prove the potential of the system to utilize the available flexibility and enhance the system robustness in reaction to the dynamics of the environment. Despite the fluctuations in the family types and quantities of the tested jobs, the cooperative agent-based scheduling employed in the proposed framework was found to significantly outperform the decentralized scheduling and approach the near optimal solution resulting from the static approach.

Work is currently ongoing in applying other priority rules and experimenting with their relative performance. In addition, developing a concept for recovering the generated schedule from deviations stemming from disturbances from the shop floor control is currently under investigation.

REFERENCES

[1] Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J. Handbook on Scheduling from theroy to applications.. Springer 2007
[2] M. P. Groover. "Fundamentals of modern manufacturing". John Wiley & Sons, New York, 2000
[3] J. Ranta, and I. Tchijov . "Economics and Success Factors of Flexible Manufacturing Systems: The Conventional Explanation Revisited". The International Journal of Flexible Manufacturing Systems, 2 (1990): 169-190
[4] R. J. Wittrock. "Scheduling Parallel Machines with Major and Minor Setup Times". The International Journal of Flexible Manufacturing Systems, 2 (1990): 329-341.
[5] K. E. Strecke "Design, planning, scheduling, and control problems of flexible manufacturing systems," Annals of Operations Research vol. 3 no. 4 pp. 1-12, January 1985.
[6] L. Wang and D. Li. A Scheduling Algorithm for Flexible Flow Shop Problem. Proceedings of the 4th World Congress on Intelligent Control and Automation. June 10-14, 2002. 3106- 3108 vol.4.
[7] J. N. Gupta. "An excursion in scheduling theory: an overview of scheduling research in the twentieth century". Production Planning & Control, Vol.13, No.2, 105-116, 2002.
[8] S. C. Graves. "A Review of Production Scheduling". Operations Research, Vol.29, No.4, Operations Management. (Jul.-Aug.,1981), pp.646-675.
[9] G. E. Vierra, J. W. Herrmann, and E. Lin. "Rescheduling Manufacturing Systems: a framework of strategies, policies, and methods". Journal of Scheduling, Vol.6, No.1, Jan.-Feb.,2003
[10] J. Hermann "Rescheduling Strategies, Policies, and Methods" in Handbook of Production Scheduling, edited by J. Hermann, Springer, 2006.
[11] R. Haupt. "A survey of priority rule-based scheduling". OR Spektrum (1989) 11:3-16.
[12] D. M. Upton. "A flexible structure for computer-controlled manufacturing systems". Manufacturing Review, 1992. 5(1):58-74. http://www.people.hbs.edu/dupton/papers/organic/WorkingPaper.html
[13] M. Wooldridge. "An Introduction to Multi Agent Systems". John Wiley & Sons, Ltd, 2002.
[14] N. R. Jennings. "An agent-based approach for building complex software systems". Communications of the ACM, 44 (4). pp. 35-41.