Monostori, L.; Egresits, Cs.; Hornyák, J.; Viharos, Zs. J.; Soft computing and hybrid AI approaches to intelligent manufacturing. Lecture Notes in Artificial Intelligence, LNAI 1416, *11th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, Castellon, Spain, 1998 jun 1-4. , pp. 763-774.

SOFT COMPUTING AND HYBRID AI APPROACHES TO INTELLIGENT MANUFACTURING

László Monostori, József Hornyák, Csaba Egresits, Zsolt János Viharos

Computer and Automation Research Institute, Hungarian Academy of Sciences
POB 63, H-1518 Budapest, Hungary
{laszlo.monostori, hornyak, egresits, viharos}@sztaki.hu

Abstract. The application of *pattern recognition* (*PR*) *techniques*, *artificial neural networks* (*ANNs*), and nowadays *hybrid artificial intelligence* (*AI*) techniques in manufacturing can be regarded as consecutive elements of a process started two decades ago. The fundamental aim of the paper is to outline the importance of soft computing and hybrid AI techniques in manufacturing by introducing a genetic algorithm (GA) based dynamic job shop scheduler and the integrated use of neural, fuzzy and GA techniques for modeling, control and monitoring purposes.

1 Introduction

The term of *Intelligent Manufacturing Systems (IMSs)* can be attributed to a tentative forecast of J. Hatvany and L. Nemes from 1978 [5]. In another landmark paper of J. Hatvany in 1983, IMSs were outlined as the next generation of manufacturing systems that - utilizing the results of artificial intelligence (AI) research - were expected to solve, within certain limits, unprecedented, unforeseen problems on the basis even of incomplete and imprecise information [4].

Soft computing technologies, like ANNs, fuzzy systems, GAs, probabilistic techniques, their combinations and their hybrid use with more traditional symbolic approaches of AI are prospective tools for realizing systems with the required behavior.

2. Dynamic job shop scheduling based on GAs

A job shop is a manufacturing production environment where a set of *m jobs* $J = \{J_1, \dots, J_m\}$ has to be performed on a set of n machines (or resources) $R = \{R_1, \dots, R_n\}$. Each job *Ji* is composed of a set of operations (or tasks) o_{ij}, $i = 1, \dots m$, $j = 1, \dots m(i)$, where *i* is the index of the job, and *j* is the index of the step (task, operation) in the overall job. In this context, tasks (operations) are regarded as scheduling entity.

The *job shop scheduling problem* involves the synchronization of the completion of *m* jobs on *n* resources, known as an *NP-hard* combinatorial optimization problem.

Scheduling is a constraint satisfaction problem where the various technological, temporal and resource capacity constraints are often ill-defined, multiple and conflicting. Several approaches have been proposed and applied for scheduling in the manufacturing domain, including linear, integer, non-linear and dynamic programming, (AI-based) heuristics, GAs, simulated annealing, dispatching rules, simulation, etc. or their combinations.

In the paper a GA-based *on-line dynamic scheduling* is presented, which is able to

- generate near optimal, valid schedules within a short time,
- react on new external and internal conditions,
- apply varying scheduling horizon, and in connection with the above issues, to
- treat alternative process plans, and
- handle multiple schedule variants at the same time.

2.1 Genetic Algorithms for job shop scheduling

The advantages of GAs have been proven in ill-behaved (such as multimodal and/or non-differentiable) and difficult to standardize problems. One of the earliest works on GAs' application to scheduling is published by Davis [1]. The representation and GA operators suggested by him are memory-intensive and provide solution for a limited set of problems giving the opportunity to improvement. Although Davis called the attention to the fact that most of real-life scheduling problems involve ill defined constraints, which are not suitable for the formal frameworks of Operations Research (OR) techniques.

In a recent work an ordinal representation is introduced by Fang et al. [3], which served as a basis for the current work by widening the standard set of constraints with constraints of dynamic scheduling (e.g. deadlines and arrival times of jobs).

Additionally, in a realistic FMS environment, some further requirements have to be taken into account, which can influence both the schedules' feasibility and optimality. These requirements include the treatment of:

- transportation times,
- set-up times which may be significant and sequence dependent,
- central and machine buffer, tool magazine and transportation capacities,
- alternative process plans for workpieces,
- operation range of the resources,
- alternative machines,
- tool life issues,
- original allocation and the occasional transportation of tools
- precedence between tasks of different jobs.

Comparing with makespan objectives of benchmark problems, schedule appraisal in a factory is usually a much more complex assignment. There are a great variety of objectives based on complete time and due date (e.g. maximum complete time or makespan, mean complete time, maximum flow time, mean flow time, maximum lateness, mean lateness, maximum tardiness, mean tardiness, number of tardy jobs

and their weighted variants) and these can be mixed with cost-revenue or utilization based fitness functions as well.

The proposed genetic algorithm is based on *ordinal representation* and *schedule builder* detailed in [10]. The representation involves information only about how to build a schedule and not about the schedule itself. As a consequence of this distinction between genetic coding and schedule generation, different constraints can be treated without any modification in the core algorithm.

The schedule builder guarantees that the schedule made by this way is *nondelay* and *legal*. The property of nondelay means that a machine immediately starts a task when it is ready to be processed and legal indicates that the schedule satisfies all constraints, for which the schedule builder was prepared.

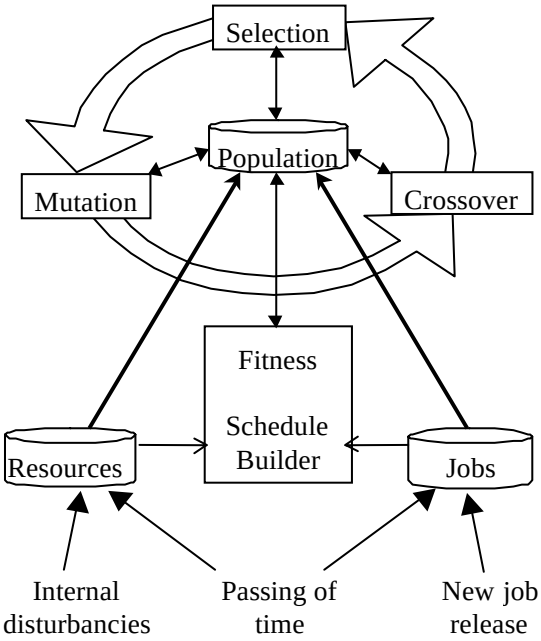

Figure 1. Functional architecture of the GA-based dynamic scheduler

2.2 On-line and dynamic aspects of the proposed solution

The GA solution presented here was planned to treat insertion new jobs in the population, passing of time and occupation of machines for a certain interval in the case of machine breakdown or repair. These events are treated while the program is continuously searching for better and better schedules (Figure 1).

Every event (passing of time, release of new jobs, delays, breakdowns, modification of deadlines or arrival times and change in fitness function) influences the individuals and their fitness values. With the exception of time passing, these events rearrange the search space, therefore, modifications in the population have to be induced

(e.g. through increased mutation rate). After every event the preservation of the appropriate part of schedule is recommended in contrast to scheduling from scratch. Consequently, it is required to work out the reactions for different events.

The main difficulties come from the time-critical nature of *dynamic on-line scheduling*. The search space, however, can be dynamically modified, i.e. decreased (as time passes, and tasks, jobs have been accomplished); increased (new jobs arrive at the system), or rearranged (e.g. at disturbances).

In the case of on-line dynamic scheduling, there is a time constraint for the scheduler for finding the best possible result. The majority of time is consumed by the schedule builder and the fitness calculation. The time required by the evaluation of different individuals of a population is nearly constant and (between linearly and quadratically) increasing function of the representation length; the search space, however, is exponentially growing with the problem size. So it is absolutely advised to decrease the space on behalf of the search efficiency. Incoming jobs with large slack time or later arrival time, can be filtered out.

Another way to improve efficiency is to filling up the initial population with appropriate individuals. If a job has later arrival time and deadline than an other has, then it is probably scheduled later and is further back in the string of an optimal individual. A guided, not uniformly distributed initialization of individuals can help GAs in these cases. The second solution can be the incorporation of built-in heuristics, which produce immediate decisions in critic situation as well.

2.3 Results

The proposed solution has been extensively tested on different benchmark and industrial problems [10].

Figure 2 and Figure 3 present an exemplary comparison of a dynamic scheduling problem. The first Gantt-diagram is the result of the problem with full information about future jobs at zero time and the second diagram shows this on-line variant, when additional jobs arrive in three points of time during scheduling.

Table 1. Start time, deadline and arrival time of the exemplary dynamic scheduling problem.

Jobs	Start Time	Deadline	Arrival Time
1, 2, 3, 4	0	500	0
5, 6	150	700	100
7, 8, 9	300	900	250
10, 11, 12, 13, 14	450	1100	400

Table 1 shows the start times, deadlines and arrival times of jobs. At arrival time the scheduler gets the information about jobs and the start time of a job means the earliest time, when the first task of this job can be started to be proceeded. The axis x

of Gantt diagrams is graded by 100 units of time. A rectangle with the number of the corresponding job indicates a reserved interval on a machine and shades help only to identify the jobs. The fitness function of GAs in both cases is the sum of lateness of jobs without any additional condition, so any schedule with zero lateness is equally appraised and considered as optima in this comparison. In Figure 2, the Gantt diagram shows an optimal schedule. In the case of on-line dynamic scheduling in Figure 3, only Job 3 (its last task is on machine 3) is delayed for 32. This delay was got after the first set of additional jobs had arrived. Originally, the optimal schedule for the first 4 jobs was reached very easily, but this early decision constrained later decisions.

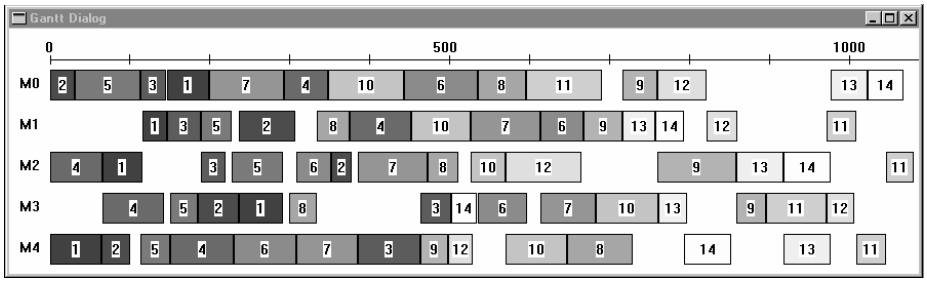

Figure 2. Gantt diagram of schedule made at zero time on the basis of full information of jobs

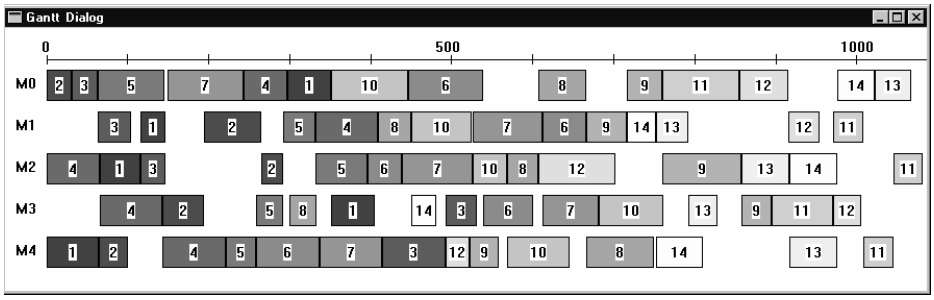

Figure 3. Gantt diagram of schedule made by on-line dynamic scheduling

As future research issues search spaces of dynamic scheduling problems, behavior and efficiency of GAs in time-critical situations will be examined. The applicability of the GA-based scheduling approach in a distributed environment is also investigated.

3. Monitoring of manufacturing processes by combining soft computing approaches

Artificial neural networks have proven to be equal, or superior, to other pattern recognition learning systems over a wide range of domains, also in cutting tool monitoring

[8]. However, the majority of ANN models (e.g. the most frequently used back propagation model) have a set of problems:

- lengthy training times,
- dependence on the initial parameters,
- lack of a problem-independent way to choose appropriate network topology,
- the incomprehensive (black box) nature of ANNs.

On the other hand, *fuzzy systems* usually do not incorporate learning ability, and it is very hard to identify fuzzy rules and tune membership functions.

3.1 Genetic algorithms for generation of neuro-fuzzy structures

In [7] a neuro-fuzzy approach was introduced and its applications in manufacturing were described. The implemented neuro-fuzzy model basically follows the main objectives of the solution described by Lin and Lee [6].

The system consists of five layers (Figure 4), the *linguistic nodes* in layers 1 and 5 represent input and output variables, respectively. Nodes in layer 2 and 4 are *term nodes* acting as *membership functions* (*MBFs*) to represent the terms of the given linguistic variable. Each neuron of layer 3 stands for one fuzzy rule (*rule nodes*). Links pointing to layer 3 define the preconditions of the rule nodes, and links between layers 3 and 4 incorporate the rules' consequences [7].

The learning algorithm of the Lin-Lee model consists of four consecutive steps [6]:

- determination of the MBFs by self-organized clustering,
- selection of the most important fuzzy rules by competitive learning,
- elimination and combination of rules,
- adjustment of MBFs' parameters by supervised back propagation learning.

In the NF model the selection of important fuzzy rules proceeds by competitive learning, requiring the generation of the whole structure at the beginning of the learning process, i.e. the net with all possible rules. This means that - at the early stage - the third layer has as many elements, as the product of the number of different MBFs assigned to the input variables. The approach leads to a combinatorial explosion, consequently, is suitable for building fuzzy logic systems with a relatively small number of linguistic variables. Another drawback to be mentioned is that the subset of "best" rules selected by the competitive phase is not necessarily the "best" subset of rules. The main goal of the solution described here is to eliminate these shortcomings by using *genetic algorithms* (GAs) for rule generation (replacing steps 2 and 3 in the above 4-step learning process).

3.2 GA representation of fuzzy rule blocks

A genetic algorithm can be used to discover a desirable optimal set of rules. In the proposed approach, instead of using a binary string, integer number string (a vector)

was used to represent a particular case or object. An individual of population is interpreted by a fuzzy rule set which is coded to a string (Figure 4).

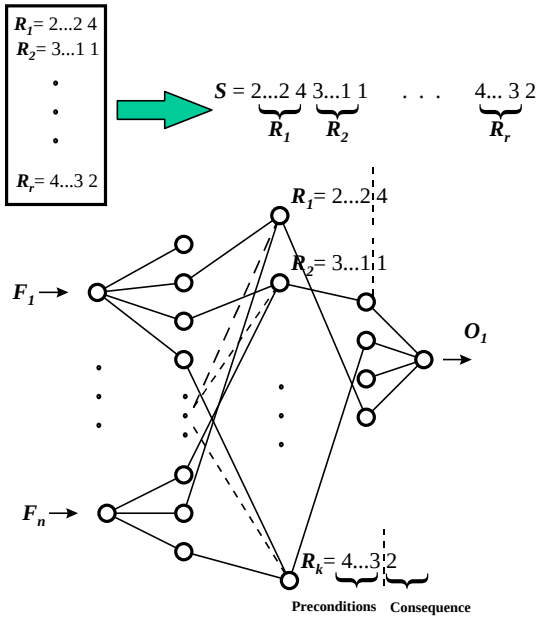

Figure 4 Representation and genetic coding of the neuro-fuzzy rules [7]

Let's consider the case with n input variables (F_1-F_n) and one output variable (O_1). A rule has the following form:

IF F_1 is *middle* and F_2 is *high* and...and F_n is *low* THEN O_1 is *weak*, where *middle*, *high*, etc. are MBFs of the corresponding variables. In the chosen coding, the MBFs of the *k*-th input or output variable are one by one mapped to natural numbers in interval [1,2,...,j_k], where j_k denotes the number of all possible MBFs of the given variable. A rule set consists of maximum *r* pieces of rules and an individual (rule set) is coded to a string:

$$S=a_{11}a_{12}...a_{1n}b_1\ a_{21}a_{22}...a_{2n}b_2\ ...\ a_{r1}a_{r2}...a_{rn}b_r\ , \qquad (1)$$

where a_{li} and b_l are the mapped values of the corresponding MBFs of the *i*-th input and the output variable, respectively, in the *l*-th rule. If, e.g., the F_1 input variable has four membership functions *very_high*, *high, low and very_low*, and F_1 is *high* in the *i*th rule, then a_{i1}=2.

A population contains strings with the same length. A special rule, named "empty" rule, is introduced in the representation without any preconditions and consequences. Every rule is of the same length, therefore, the empty rule is coded with *n*+1 (number

of inputs + number of outputs) “0” characters. Spreading of empty rules results in decreasing the number of rules in a rule set.

Most of NF models operate with rules having all preconditions, and generating rule set with incomplete rules or simplifying and reducing rule sets cause difficulties. In this representation introduction of incomplete rules is a trivial task. “0” characters are simply allowed in place of any a_{Ii}. As a further benefit, this solution substantially expands the search-space.

Usual genetic operators (single and multipoint crossover and mutation) are used in the new approach. For fitness calculation the system has to generate the rules for the net and run a forward process. In case of classification problems, the fitness value was chosen to be a function of the weighted quadratic error of the training patterns using the actual rule set, the recognition rate of the system (in case of estimation process this factor is 1.0), the length of the rules and the size of the network:

$$Fit(net) = \left(\sum_{i=1}^{N} \sum_{j}^{K} (o_{ij} - t_{ij})^2 \right)^{-1} * R * \eta_1 * \eta_2 , \qquad (2)$$

where N is the number of patters in the training set, K is the number of outputs of network, R is a monotonously increasing function of the recognition rate of the system, η_1 measures the length of rules (shorter rules are better) and η_2 measures the size of the network (smaller network is better).

3.3 Experiments

One the one hand, some well-known benchmark problems (Iris, Spiral of Archimedes, XOR) where the properties of the data are known well and which are used in various papers to illustrate the capabilities of unsupervised and supervised systems. The described neuro-fuzzy algorithm resulted in comparable performance, eliminating some shortcomings of the BP approach enumerated in the introduction, however, sometimes with lower recognition rate for training patterns [2].

Other investigations referred to the state classification of milling tools. Given 4 wear classes (sharp tools, tools with an average wear of teeth of 0.25 0.45 mm, respectively, and tools with broken (missing) insert), the task was to generate and compare ANN and NF structures able to reliably classify unknown patterns characterizing different wear states.

Using the Lin-Lee model, after the initialization phase, taking all the possible combinations, according to the 4, 3, 2, 3, 4, 4 MBFs assigned to the 6 input linguistic variables, 4*3*2*3*4*4 = 1152 rule nodes were generated. During the competitive learning phase 1138 (!) of them were deleted, resulting in a network structure with 14 rules. This *self-organization* is a very important feature of the chosen model. The number of eliminated links was 4594 [9].

Applying genetic techniques, the algorithm generally offers solutions with shorter and simpler rule sets and results in comparable recognition performance as the investigated previous NF approaches [2]. The introduction of empty rules gives chances to

decrease the number of rules helping to evolve the optimal subset of rules. The permission of incomplete rules offers simpler rule sets. Comparing the two neuro-fuzzy techniques, in genetic model the algorithm eliminated further 32 precondition links (Table 2), in contrast to the implemented version of the Lin-Lee model where the rules are fully connected in precondition site.

Table 2 The set of rules remained after genetic learning [2]

	F1	**F2**	**F3**	**F4**	**F5**	**F6**	**Outp.**
1	high	high	--	--	high	--	Wear025
2	high	--	--	--	med	--	Wear025
3	high	--	--	--	high	--	Wear045
4	m_high	--	--	high	med	m_high	Wear045
5	m_high	m_low	med	--	high	m_high	Wear045
6	m_high	m_low	med	--	med	m_high	Wear045
7	m_low	--	low	--	med	m_low	Broken
8	m_low	low	--	low	--	m_low	Broken
9	low	low	--	low	--	m_low	Broken
10	m_high	m_low	med	--	med	low	Broken
11	m_low	--	low	high	--	--	Sharp
12	m_low	low	--	low	--	low	Broken
13	m_low	--	low	low	--	low	Broken
14	low	low	low	--	--	low	Sharp

As a conclusion, NF systems as hybrid learning systems can comply with the fundamental requirements of intelligent manufacturing, namely *real-time nature, combined structure and parameter learning ability, handling of uncertainties, managing and learning both symbolic and numerical information*. This approach has the additional benefits of *explicit knowledge representation*, which is easily *modifiable*, and incorporates a kind of *explanation facility*. Its structure is potentially applicable for integrating *pattern-based*, *rule-based* and *analytical knowledge*, which integration is of fundamental importance in different manufacturing assignments.

A genetic algorithm-based approach was introduced to overcome problems experienced during applications of previous NF algorithms in manufacturing. The first results are promising. There are, however, numerous open problems also in this narrow field, e.g. the automatic generation of the number of MBFs assigned to input and output variables, the automatic or computer-aided partitioning and training of more complex assignments, etc., which are subjects of further research.

4. Acknowledgments

This work was partially supported by the *Nat. Res. Foundation, Hungary,* Grant Nos. F023628 and T026486. A part of the work was covered by the *Nat. Comm. for Techn. Dev., Hungary* Grants (EU-96-B4-025 and EU-EU-97-A3-099), which promote Hun-

garian research activity related to the *ESPRIT LTR Working Groups* (IiMB 21108 and IMS 21995).

5. References

1. Davis, L., Job shop scheduling with genetic algorithms, in Proc. of the International Conference on Genetic Algorithms and their Applications, Pittsburgh, Morgan Kaufman, (1985) 136-140.
2. Egresits, Cs.; Monostori, L.; Hornyák, J., Multistrategy learning approaches to generate and tune fuzzy control structures and their applications in manufacturing, Proceedings of the Second World Congress on Intelligent Manufacturing Processes and Systems, June 10-13, Budapest, Hungary, Springer, (1997) 88-94, also in Journal of Intelligent Manufacturing, Special Issue on Soft Computing Approaches to Manufacturing, Chapman and Hall, 1998, (in print)
3. Fang, Hsiao-Lan; Ross, P.; Corne, D., A promising Genetic Algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems, Proc. of the Fifth International Conference on Genetic Algorithms, San Mateo, Morgan Kaufmann, (1993) 375-382.
4. Hatvany, J., The efficient use of deficient information, CIRP Annals, Vol. 32/1, (1983) 423-425.
5. Hatvany, J., Nemes, L.,. Intelligent manufacturing systems - a tentative forecast, In: A link between science and applications of automatic control, Proc. of the VIIth IFAC World Congress, (A. Niemi, (Ed.)), June 12-16, Helsinki, Finland, Vol. 2, (1978) 895-899.
6. Lin, C. H.; Lee, C. S. G., Neural-network-based fuzzy logic control and decision system, IEEE Trans. on Comp., Vol. 40, Dec., (1991) 1320-1336.
7. Monostori L.; Egresits Cs.; Kádár B., Hybrid AI solutions and their application in manufacturing, Proc. of IEA/AIE-96, The Ninth Int. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, June 4-7, 1996, Fukuoka, Japan, Gordon and Breach Publishers, (1996) 469- 478.
8. Monostori, L., A step towards intelligent manufacturing: Modeling and monitoring of manufacturing processes through artificial neural networks, CIRP Annals, 42, No. 1, (1993) 485-488.
9. Monostori, L.; Egresits, Cs., On hybrid learning and its application in intelligent manufacturing. Preprints of the Second Int. Workshop on Learning in IMSs, Budapest, Hungary, April 20-21, (1995) 655-670, and Computers in Industry, Special issue on Learning in IMSs, Vol. 33, 1997, pp. 111-117.
10. Monostori, L.; Hornyák, J.; Kádár, B., Novel approaches to production planning and control, IMS Europe 1998, April 15-17, Lausanne, Switzerland (in print)
11. Monostori, L.; Márkus, A.; Van Brussel, H.; Westkämper, E., Machine learning approaches to manufacturing, Annals of the CIRP, Vol. 45, No. 2, (1996) 675-712.