CHAPTER 11

Using neural networks for reactive scheduling

M. Garetti[a] and M. Taisch[b]

[a] Professor of Industrial Technology
Dipartimento di Econornia e Produzione – Politecnico di Milano, 20133, Milano, ITALY

[b]research fellow of AS. SC. UN. (Associazione Scuole Universitaire) c/o Politecnico di Milano, sede di Lecco, 22053, Lecco, ITALY

Keyword Codes: I.2.1, I.2.8
Keywords: Applications and Expert Systems; Control Methods and Search

Abstract

In the recent past, neural network architectures have been developed to solve several optimization problems including scheduling In this paper, after describing the state of the art in the use of neural networks for scheduling purposes, the potential benefits and limitations of their application to reactive scheduling will be discussed. After that an alternative approach to reactive scheduling, based on the use of ANNs in hybrid architectures, is proposed. This should make it possible to overcome the problems currently limiting the application of ANNs to reactive scheduling.

1. INTRODUCTION

As known, scheduling is a constraint-based optimization problem aimed at detecting the right job sequences for each available resource while at the same time minimizing a cost function. The complexity of the problem is well known and can be traced back to the need to meet both soft and hard constraints, such as due dates and operations sequences.

Several techniques, i e mathematical and statistical models, dispatching rules, knowledge-based systems, look-ahead algorithms, heuristic algorithms and genetic algorithms have been used in the past to solvè scheduling problems. These techniques have led to the development of increasingly better solutions, without obtaining completely satisfactory ones due to the natural complexity of scheduling problems.

So, the ability of some Artificial Neural Networks (ANNs) architectures to solve optimization problems makes them a promising technology for scheduling purposes, stimulating the interest of many authors in this topic. Moreover, the massively parallel and interconnected structure of neural networks enables them to provide quick solutions, thus making them the ideal candidate for reactive scheduling applications.

For example, let's take the most widely used neural network model - i.e. the Hopfield network developed by Hopfield and Tank [2, 7]. This network could be either discrete (the status of neurons is zero or one) or continuous (every value between zero and one is acceptable). The solving algorithm gives neurons the values required to minimize a cost-energy function whose

R. Kerr et al. (eds.), *Artificial Intelligence in Reactive Scheduling*

terms describe every single constraint of the scheduling problem. This way the scheduling problem can be solved.

In this paper, the authors will first describe the state of the art of ANNs as a scheduling tool, and their potential benefits and drawbacks when applied to reactive scheduling. Then an alternative approach to the use of ANNs for reactive scheduling, based on hybrid architectures is suggested. In it, ANNs are integrated with other tools and only used to address those areas where the application of ANNs is regarded as most promising.

2. STATE OF THE ART IN THE USE OF ANNs FOR SCHEDULING

Scheduling is called *predictive* when it aims at finding an optimal combination of job sequences for a set of available resources (while at the same time minimizing a cost function). But in a real shop-floor environment, where a number of unexpected events invalidating current schedules often occur, a revising process is needed to update the schedule and to develop a new feasible and optimal solution. This latter approach to the scheduling problem is called *reactive scheduling.*

In the recent past, the use of ANNs as a tool to solve predictive scheduling problems has been successfully investigated. In addition, the application of ANNs to different production contexts demonstrated the flexibility of this technique.

Even though the papers we will consider afterwards deal with the application of ANNs to general scheduling problems, they can be considered as a starting point to evaluate the ability of ANNs to solve reactive scheduling problems.

The first papers suggesting the use of ANNs to solve scheduling problems were those by Foo and Takefuji [1] dealing with a 4-job, 3-machine job-shop, and by Zhou, Cherkassky, Baldwin and Hong [12] with a 20-job, 20-machine job-shop. By applying them to problems with a known solution, the authors demonstrated that ANNs never failed to find the right solution.

The presence of constraints has been investigated by many authors. Van Hulle [9] successfully reformulated the 4-job, 3-machine problem as a goal-programming problem. Thawonmas, Shiratori, and Noguchi [8] set up a problem with deadline constraints and solved it by using two different integrated ANNs.

Task priorities were investigated by Zhang, Yan and Chang [11], using a scheme suggested by Zhou *et al.* Priorities were analysed by a cost-energy function through the introduction of a coefficient whose value was low if the constraint was met and high if priority needs were not complied with. This approach makes it possible to deal with any kind of constraints.

The first application of ANNs to a real case was suggested by Khaw, Lim and Lim [3]. Their paper describes the use of a hybrid neural network for the scheduling of a group technology manufacturing cell for the production of submersible pumps in a make-to-order configuration. A first recurrent neural network is used to generate all feasible solutions, after that the solutions are evaluated by a second network and the best one is chosen.

A different kind of application was suggested by Rabelo, Alptekin, Kiran [6], who used a neural network as a rule evaluator rather than as an optimizer. After receiving information from a job database and a due-date database, the ANN picks a rule from a set of available dispatching rules (Shortest Process Time, Earliest Due Date, Critical Ratio, Slack Time Remaining, Slack/Operation, Least Work Remaining). The chosen rule is then passed to an

expert system scheduler to be turned into a solution. This approach shows how two different AI disciplines can be integrated to solve a task needing more problem-solving methods.

3. POTENTIAL BENEFITS AND DRAWBACKS OF THE APPLICATION OF ANNs TO REACTIVE SCHEDULING

The two most important features which make ANNs a promising technique to solve reactive scheduling problems are:

- their approach to the problem is not constructive (i.e. no written rules are needed to communicate how solutions should be implemented), but rather descriptive (i.e. only the modelling of constraints and priorities is needed);
- computing time is very low compared to other scheduling techniques, due to the massive parallel interconnections of ANNs.

The combined use of these features makes ANN's a knowledge-based alternative or an add-on to the traditional tools used for reactive scheduling.

The performance of ANNs in reactive scheduling depends on the specific production context they are applied to, as well as on the implementation method adopted. Two different approaches are possible:

- use of ANNs to generate a feasible solution;
- use of ANNs to evaluate the scheduling sequences generated by other techniques.

In the former case the ANN is the basic element generating the schedule, whereas in the latter it is just an element of a hybrid scheduling system. In the former case an optimization neural architecture is used, whereas in the latter one the architecture depends on the function addressed by the ANN.

3.1. Using stand-alone ANNs for reactive scheduling

The potential benefits of ANNs used as a stand-alone tool in reactive scheduling appear to be as follows:

- short computing time vis-à-vis traditional techniques;
- high reconfiguration flexibility when the production context is changed;
- no need to develop an algorithm to solve the scheduling problem thanks to the distributed knowledge of ANNs.

Despite these benefits, the use of ANNs as a stand-alone tool still has some drawbacks when dealing with reactive scheduling. Further constraints, raised by current production schedules, pile up on top of the usual constraints of predictive scheduling, making even more difficult to include such constraints in the reliable analytical model required to implement Hopfield's network.

3.2. Hybrid neural architectures for reactive scheduling

In the light of these considerations, the authors intend to suggest an approach to reactive scheduling based on the combined use of ANNs with more traditional tools, i.e. building hybrid architectures for scheduling (where "hybrid" means that these architectures imply the integrated use of different tools including ANN, Expert Systems, Genetic Alghoritms, etc.). This approach is suggested following the complexity of the reactive scheduling problem and

the need to break it down into easier sub-problems. So each sub-problem can be solved by a specific tool, thus paving the way for the use of a higher number of solving methods. The architectures described hereafter are to be regarded as preliminary suggestions only; their sole purpose is to show the potential efficiency of ANNs in solving reactive scheduling problems and they are not intended as exhaustive solutions.

4. HYBRID ARCHITECTURE 1: COMBINING NEURAL NETWORKS, SIMULATION AND GENETIC ALGORITHMS

The first architecture we will discuss here was suggested by Jones and Rabelo [4] in a paper about predictive scheduling (Figure 1). It is a hybrid architecture which makes use of ANNs, simulation tools and genetic algorithms for predictive scheduling purposes. However, the authors think it might be applied to reactive scheduling as well after undergoing some changes discussed later.

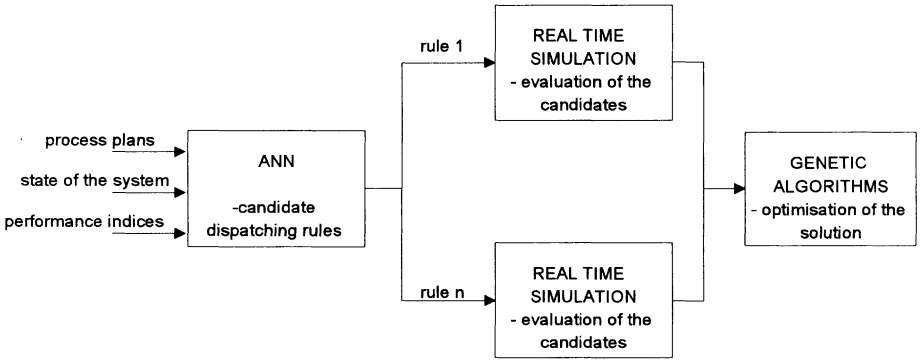

Figure 1. Hybrid architecture for reactive scheduling combining ANN, simulation and genetic algorithms. (adapted from Jones, A.T., Rabelo, Luis C., Real-Time Decision Making Using Neural Nets, Simulation, and Genetic Algorithms, Int. J. of Flexible Automation and Integrated Manufacturing, Vol.1, No. 2, 1993)

The original approach is made up of three steps: i) detection of a set of dispatching rules which appear to be the best candidates for solving a specific problem, ii) evaluation of the rules and selection of one of them, iii) and finally optimization of the resulting solution.

In *Step 1* a set of dispatching rules is selected with the help of a neural network. The network is supplied information about plant status, working plans and performance indexes (i.e. indexes measuring the goals to be maximised or minimised).

The plant status is described by data about the jobs included in the system, their location (buffers, machines, feeding systems), their due dates and expected release dates, machine workloads, ect. Working plans are described by working times and routing plans. The ANN is also supplied with performance indexes like average delay, job flow time, machine saturation rate, etc.

This information has to be transalated into a numeric representation the neural network can understand. The creation of a significant knowledge-base for the ANN learning process can be

based on specifically developed and selected examples or on a set of past scheduling cases. Obviously, the efficiency of the tool in selecting "good" candidate rules depends on the significance of the learning set adopted. On the other hand, it is also possible to resort to self-learning tools during field use to increase ANN performance.

Once a set of candidate rules has been selected, an evaluation of the actual impact of each rule on the performance of the manufacturing system is carried out through a series of simulations in *Step 2*. This evaluation is based on discrete-event simulation, i.e. by having each rule generate its own scheduling plan. The initial status of the simulation model corresponds to the plant status when rescheduling becomes necessary. The rule capable of generating the best scheduling towards assigned goals is chosen for the development of the scheduling plan.

Optimization is carried out during *Step 3*. Since no rule can maximise (or minimise) all existing goals but rather a compromise solution among all performance criteria has to be achieved, a further optimization of the scheduling plan is possible. To do this, the authors used the third tool making up their architecture, i.e. genetic algorithms. Through a repetitive process including the generation and testing of alternative solutions, this tool looks for local optimizations of the scheduling plan while complying with existing constraints.

The use of this architecture for reactive scheduling has its critical point in simulation, which might not guarantee the required reactivity. An alternative approach (Figure 2) replaces simulation with a plant emulator based on a supervised-learning ANN. The input data for this neural simulator are provided by a set of plant-status descriptors and by the rule currently being tested, with plant performance data as the resulting output. This approach would worsen output data reliability (due to the approximation process carried out by the neural simulator) but it would decrease response times, which in turn would make its use efficient in reactive scheduling applications.

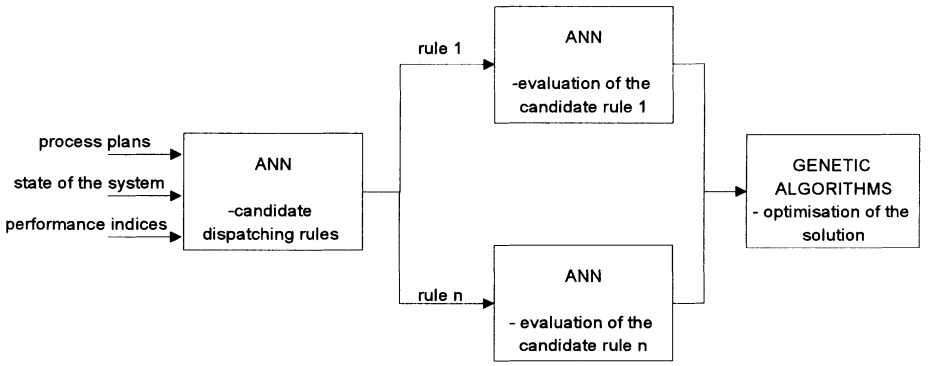

Figure 2. Hybrid architecture 1 revised for reactive scheduling with a neural simulator .

5. HYBRID ARCHITECTURE 2: COMBINING ANNs AND GENETIC ALGORITHMS

The model described above used genetic algorithms only for the local optimization of a solution developed by another tool (i.e. the dispatching rule). In this section, the authors are going to discuss another hybrid architecture which uses genetic algorithms to generate a final solution.

As is well known, the solutions of a scheduling problem can be generated constructively (i.e. through the use of repetitive logics) or following a combined approach (i.e. combining job allocations among all the machines making up the line). Genetic algorithms fall into the latter scheduling category, since their operation includes the following steps:

1. system initialization with a random chromosome set;
2. chromosome evaluation according to a fitness function;
3. chromosome selection based on the outcome of the fitness function;
4. reproduction, crossover and mutation;
5. steps 2 through 5 are repeated until no more significant performance increases are detected.

This description shows that, apart from the reproduction, crossover and mutation rules adopted, the evaluation function (i.e. the fitness function) plays a major role in guaranteeing a fast and efficient algorithm convergence. Several examples found in literature use single-goal fitness functions, whereas conflicting goals are usually pursued in real-life scheduling applications. Thus, real-life requirements are better met by multiple-goal functions which, though being more complex, are capable of evaluating several conflicting aspects of the problem to be solved. However, it is extremely difficult to translate goal relationships and their relative weight into analytical models.

Based on these remarks, the use of supervised-learning ANNs - whose reliability as a multiple-goal evaluation tool was demonstrated above - is suitable due to their architectural features (Figure 3).

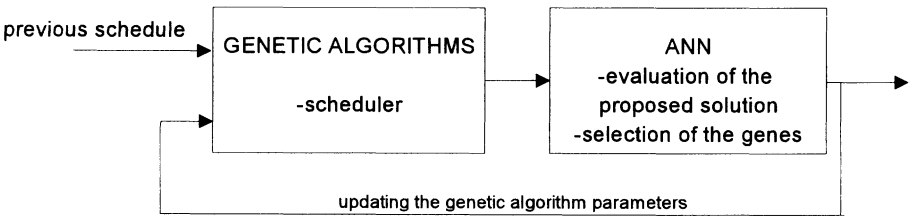

Figure 3. Hybrid architecture 2: ANNs are used as a multi-criteria fitness function for GA

6. HYBRID ARCHITECTURE 3: COMBINING ANNs WITH AN EXPERT SCHEDULER

Among other things, ANNs can also carry out system diagnostic functions, since they are able to represent the function leading from symptoms back to causes. The benefits of using

ANNs, rather than other techniques, can be found in the self-generation of knowledge, which does not require an explicit description of the I/O link.

On the other hand, some scheduling techniques (i.e. dispatching rules, expert systems, ect.) require explicit values for some tuning parameters of their operating logic to function correctly, and this strongly affects the quality of the solutions found by the scheduler.

This makes it necessary to estimate these parameter values, which depend on countless factors based on their synthetic nature: in general terms, they depend on the nature and status of the system (jobs making up the system, due dates, current machine workloads, batch policies, priorities, etc.), on working plans (working times and routing plans) and on goals (flow-time minimisation, makespan minimisation, machine saturation maximisation, average delay minimisation, late job minimisation, etc.).

As Figure 4 shows, their features as universal function regressors make ANNs good candidates for this task.

Moreover, this architecture makes a self-learning process possible, since the scheduler's performance during rescheduling can be used to broaden the ANN learning set, thus allowing a more extensive learning and improving the scheduler's performance.

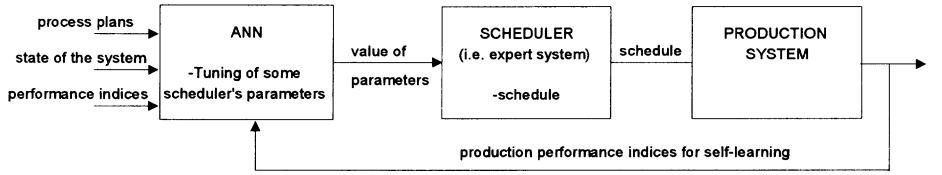

Figure 4. Hybrid architecture 3: an ANN is used to detect a set of tuning parameters for the expert scheduler.

7. HYBRID ARCHITECTURE 4: NEGOTIATION-BASED COOPERATIVE CONTROL AND SCHEDULING

In the last few years, research in negotiation-based cooperative control and scheduling has been carried out by many authors. In this section, the authors will discuss how ANNs can be used within this framework to build an effective negotiation-based reactive scheduling system.

A negotiation-based cooperative system is made up of autonomous agents interacting with one another [5,10].

An autonomous agent is an intelligent functional unit modeling parts or resources in the shop-floor environment. The agents communicate and negotiate with other agents in order to achieve a near-optimal solution for the scheduling problem. This negotiation is usually carried out according to a "market-like" model (game theory rules are considered too). Each time an event occurs, the shop floor reacts by restarting the negotiation process, which involves all agents affected by that event. Therefore, this approach results in a reactive scheduling system.

Two features seem to play a critical role in the performance of this scheduling framework (Figure 5): first, the 'market laws' to which agents have to refer, have an impact on the global performance of the system and can result in different control strategies and in the fulfilment of different system goals; secondly, agents need to have objective functions to comply with.

These functions are likely to be multiple-goal evaluation functions (i.e. minimization of flow time and production costs, etc.) depending on the external conditions and on the system status.

Within this framework, ANNs can effectively and efficiently perform both of the above-mentioned tasks.

The first problem implies the generation by the supervising system of a price and rule framework based on the overall goals of the production system and on the control strategy to be adopted. The price and rule framework can be represented by a vector of parameters, the values of which represent the relative weight or importance of each overall goal (e.g. minimising flow times is more important than maximising parts quality). The authors think ANNs should be an ideal tool to use in this case due to the complex relationships existing between inputs (the overall goals) and outputs (prices and rules). A second issue supporting this choice is the possibility of a self-learning process based on the measurement of system performance.

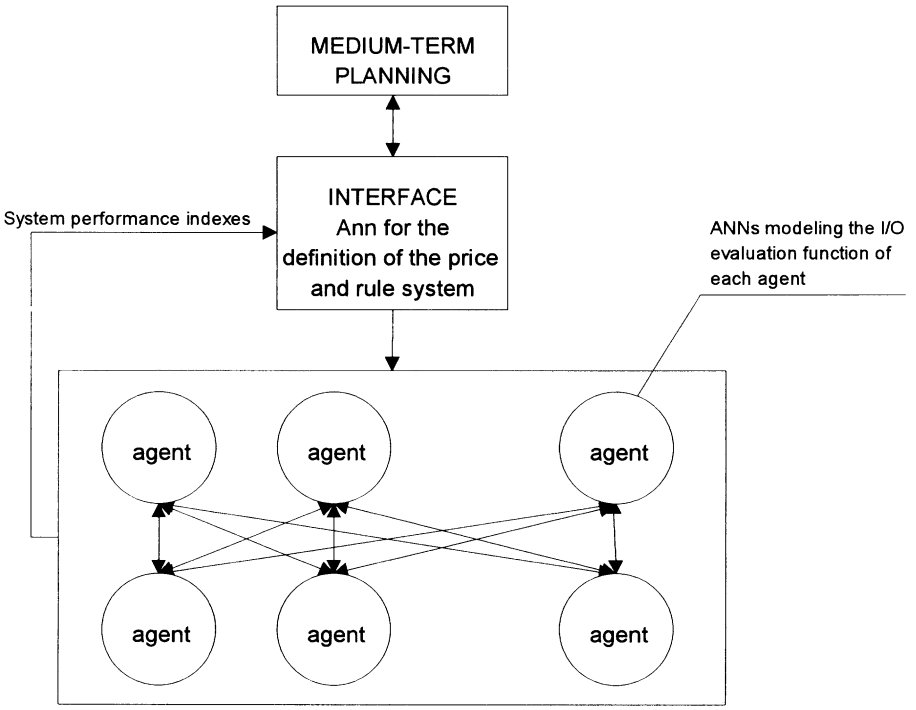

Figure 5. Hybrid architecture 4: ANNs are used for negotiation-based cooperative control and scheduling.

The second critical aspect of a well-performing negotiation-based coordinative control and scheduling system is the "intelligence" of independent agents, where by intelligence we mean the agent's ability to represent its goal set, priorities, constraints, alternative process plans, etc in an evaluation function.

The evaluation function is a basic component of a market-like model, since it is the tool which allows the negotiation process to take place inside each agent. Thanks to this function, parts can find the most convenient resource based on job goals, job priority, etc., whereas resources can adjust their own price to accept the right parts. Moreover, an independent agent can improve its bargaining ability thanks to the experience gathered during past negotiations.

For these reasons, the authors think a supervised-learning ANN is a suitable tool for the modeling of the behaviour of independent agents.

8. FURTHER AREAS OF INVESTIGATION

In spite of the proven capability of ANNs to successfully solve scheduling problems, a lot of investigation is still needed. In the authors opinion, interesting research-work to be done deals with the following topics:

- application of ANNs to automated production systems in which scheduling reactiveness is fundamental to avoid plant stops;
- application of ANNs to highly turbulent market situations, in which reactive scheduling enables production systems to meet demand;
- application of ANNs to complex production systems with a high number of jobs and machines, where the poor sensitivity of ANNs to dimensional problems (ANNs are parallel computing structures) allows to control computing times and hence scheduling reactiveness.

The above-mentioned topics could be investigated with two different approaches: use of ANNs as a 'stand-alone' tool, in the first place and integration of ANNs into complex scheduling systems, in the second one.

9. CONCLUSIONS

Many researchers have demonstrated the capability of ANN's to solve general scheduling problems when used as a stand-alone tool. On the other hand, some features of ANN technology make them very promising for reactive scheduling purposes, but future research must get a deeper knowledge of their behaviour in reactive scheduling applications.

Due to the complexity of reactive scheduling problems, a single neural tool does not seem to have the required effectiveness. Therefore, an alternative approach to the use of ANNs has been presented in this paper in the shape of hybrid neural architectures involving different techniques, with every single phase of the solution being carried out by a specific tool. Future research work will address the implementation of some of the proposed architectures.

REFERENCES

1. Foo, Y.P. and Takefuji,Y., Stochastich neural network for solving job sho scheduling, Proc. of the IEEE 2nd International Conference on Neural Networks, San Diego, June 1988.
2. Hopfield, J.J., Tank, D.W., Neural Coputation of Decisions in Optimization Problems, Biol. Cybernetics, Vol. 52, 1985, pp. 141-152.
3. Khaw, J.F.C., Lim, B.S., Lim, L.E.N., Scheduling a Group Technology Manufacturing Cell Using a Hybrid Neural Network, Int. Journal of Flexible Automation and Integrated Manufacturing , Vol. 1, No. 2, 1993, pp.105-117
4. Jones, A.T., Rabelo, Luis C., Real-Time Decision Making Using Neural Nets, Simulation, and Genetic Algorithms, International Journal of Flexible Automation and Integrated Manufacturing, Vol. 1, No. 2, 1993, pp.119-131.
5. Lin, G.Y.J., Solberg, J.J., Integrated Shop Floor Control Using Autonomous Agents, Ile Transactions, 1992, Vol. 24, No. 3, pp. 57-71.
6. Rabelo, L. C., Alptekin, S., Kiran, A. S., Synergy of Artificial Neural Networks and Knowledge-based Expert Systems for Intelligent FMS Scheduling, Proc. of the IJCNN, San Diego, CA.,1990, Vol. 1, pp. 359-366.
7. Tank, D.W., Hopfield, J.J., Simple Neural Optimization Networks-an A/D Converter, Singnal Decision Circuit, and a Linear Programming Circuit, IEEE Trans. Circuits Syst., Vol. CAS-33, No.5, 1986, pp. 533-541.
8. Thawonmas, R., Shiratori, N., Noguchi, S., Real-Time Neural Network Scheduler, Proc. of the IJCNN, Beijing, China, 1990, Vol. 1, pp. 391-396.
9. Van Hulle, M. M., A Goal Programming Network for Mixed Integer Linear Programming: a Case Study for the Job-Shop Scheduling Problem, International Journal of Neural Systems, Vol. 2, No.3, 1991.
10. A.Villa, Decentralized production scheduling by a network of local intelligent controllers, in Proc 10th Int. Conf. on Computer Aided Production Engineering,1994
11. Zhang, C. S., Yan, P.F., Chang, T., Solving Job-Shop Scheduling Problem with Priority Using Neural Network, Proc. of the IJCNN, 1991, Singapore, pp.1361-1366.
12. Zhou D.N., Cherkassky V., Baldwin T.R., Hong D.W., Scaling Neural Networks for Job-shop Scheduling, Proc. of the IJCNN, San Diego, CA, Vol 3. 1990, pp. 889-894.