JOB-SHOP SCHEDULING USING NEURAL NETWORKS[1]

A. S. Jain† • S. Meeran

Department of Applied Physics and Electronic and Mechanical Engineering, Dundee University, Dundee, Scotland, UK, DD1 4HN

A.Z.Jain@dundee.ac.uk • s.meeran@dundee.ac.uk

†The author whom correspondence should be addressed to

Abstract. *Complete enumeration of all the sequences to establish global optimality is not feasible as the search space, for a general job-shop scheduling problem,* Π_G *has an upper bound of* $(n!)^m$*. Since the early fifties a great deal of research attention has been focused on solving* Π_G*, resulting in a wide variety of approaches such as Branch and Bound, Simulated Annealing, Tabu Search, etc. However limited success has been achieved by these methods due to the shear intractability of this generic scheduling problem. Recently, much effort has been concentrated on using neural networks to solve* Π_G *as they are capable of adapting to new environments with little human intervention and can mimic thought processes. Major contributions in solving* Π_G *using a Hopfield neural network, as well as applications of back-error propagation to general scheduling problems are presented. To overcome the deficiencies in these applications a modified back-error propagation model, a simple yet powerful parallel architecture which can be successfully simulated on a personal computer, is applied to solve* Π_G*.*

KEYWORDS :- JOB-SHOP; SCHEDULING; *NP*-HARD; NEURAL NETWORKS; MODIFIED BACK-ERROR PROPAGATION; SUPERVISED LEARNING; INPUT / OUTPUT MAPPING

[1] Published in the *International Journal of Production Research*, May 1998, **36**(5), 1249-1272.

1. INTRODUCTION

With the emphasis on zero inventory and shorter product life cycles, the need for efficient and accurate schedules becomes increasingly acute. Scheduling is essentially concerned with solving a Constraint Optimisation Problem (*COP*) and in the context of manufacturing it involves finding a sequential allocation of competing resources that optimises a particular objective function, *f(x)*, to give *f(x)**, subject to certain constraints. Φ is the set of equality constraints in the problem and Ω is the set of inequality constraints; both sets are finite. $\Re^n$ is the feasible region in *n*-dimensional space with the membership of sequences of resources, $\{x_q\}$, $q = 1, 2, ..., m$. Each x_q is associated with a makespan which has to be smaller or equal to the known upper bound to qualify for the membership.

$$f(x)^* = minimise\, f(x), \quad x \in \Re^n$$
$$subject\ \ to$$
$$c_i(x) = 0, \quad \in \Phi, \quad 1 \le i \le n$$
$$c_j(x) \ge 0, \quad \in \Omega, \quad 1 \le j \le n$$

The manufacturing processes required to produce a typical product, M_p, consists of a set of different factory operations F_o where $F_o \subseteq X$, X is the set of all *n* resources in the factory $\{x_1, x_2, ..., x_n\}$, $x_i = \{y_{i1}, y_{i2}, ..., y_{im}\}$ and *y* is the set of *m* machines in each resource. The aim is therefore to find the optimum sequence of factory resources for M_p, F_o* which corresponds to *f(x)** such that M_p can be produced in the fastest time possible. For small batches of M_p the difference between *f(x)** and $f(x)^L$*, a suboptimal or local minima, is negligible. However as the production size of M_p increases the difference $f(x)^* - f(x)^{L^*}$, even if it is one second per product, may become significant. Therefore finding the optimum sequence is of paramount importance. Smit (1992) specifies several basic material flow routes by which the products move in a manufacturing environment, the most common being the flowshop, *F*; permutation flowshop *P*, where passing is not allowed, and the job-shop *J* or *G* (the difference is described

below). In a job-shop F_o is different for each M_p while in a flowshop each product has the same F_o.

Job-shop scheduling (*JSS*), as the most general of the classical scheduling problems, has generated a great deal of research (Johnson 1954, Conway *et al.* 1967, Blackstone *et al.* 1982, Adams *et al.* 1988, Carlier and Pinson 1989, Applegate and Cook 1991, Dell'Amico and Trubian 1993, Foo *et al.* 1995, Yamada and Nakano 1996a, b). In job-shop scheduling the objective might be to minimise the makespan or maximise machine utilisation, subject to constraints such as the number of machines, plant capacity, labour availability, etc.

The job-shop scheduling problem, Π_G, consists of a set of m machines $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m\}$, and a collection of n jobs $\{\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_n\}$ to be scheduled, where each job must pass through each machine once only. Each job has its own processing order and this may bear no relation to the processing order of any other job. Often technological constraints demand that each job should be processed through the machines in a particular order (precedence constraints). This is called a *J* job-shop (Conway *et al.* 1967). However for some job-shops there are no restrictions on the form of technological constraints. They are known as *G* job-shops (Conway *et al.* 1967) or open shops. In addition a class of models known as dag (*D*) or mixed shops (Shmoys *et al.* 1994) arise such that only a partial precedence ordering is given. Notationally this is summarised as :-

$$P \subseteq F \subseteq J \subseteq D \subseteq G$$

It is evident that in Π_G each job consists of m subjobs, called operations, with one machine for each subjob. Since there are n jobs, each machine must perform n operations, so there are $n!$ possible sequences on each machine. If these $n!$ sequences are independently chosen for each machine, then there are $(n!)^m$ possible solutions. Thus a problem consisting of 20 jobs and 10 machines (20×10) has 7.2651×10^{183} possible solutions in principle. However some of these will not be feasible due to precedence and disjunctive constraints. Nevertheless

complete enumeration of all the feasible solutions to identify the optimal one is not practical. Due to this factorial explosion, Π_G falls into a large class of intractable numerical problems known as $\mathcal{NP}$-Hard ($\mathcal{NP}$ stands for non deterministic polynomial). $\Pi_G \in \mathcal{H}$ and $\mathcal{H} \subseteq \mathcal{N} \subseteq \mathcal{P}$, where $\mathcal{H}$ is the set of $\mathcal{NP}$-Hard problems, $\mathcal{N}$ is the set of $\mathcal{NP}$ problems, both sets are finite and constrained within the infinite set of numerical problems, $\mathcal{P}$. The computational complexity in polynomial problems is given as $O(n^m)$, where n is the size of the problem and m is the degree of the polynomial while in $\mathcal{NP}$-Hard problems it is $O(a^n)$ (Cook 1971, Garey and Johnson 1979). Conventional approaches are therefore only useful for trivial instances. Heuristic search techniques can be used to solve larger problems but they often forego guarantees of an optimal solution for gains in speed. In recent years the technological advancements in hardware and software have encouraged new application tools such as neural networks to be applied to combinatorially exploding $\mathcal{NP}$-Hard problems, in particular Π_G (Foo and Takefuji 1988a, b, c; Zhou *et al.* 1991; Yih *et al.* 1991; Potvin *et al.* 1992; Chang and Nam 1993; Willems and Rooda 1994; Kim *et al.* 1995; Sabuncuoglu and Gurgun 1996).

Here we collate and review major works in solving Π_G using neural networks. We describe the scheduling systems in two major categories: one using the Hopfield method the other using back-error propagation (*BEP*). The paper is organised as follows: after introducing neural networks a review of the main research on Hopfield neural network based scheduling is given. The paper then goes on to explore the scheduling systems that use a *BEP* paradigm and specifically highlights the benefit of using a modified *BEP* structure in minimising the makespan for the job-shop problem.

2. NEURAL NETWORKS MODELS APPLIED TO Π_G

Numerous approaches have been used to solve Π_G. Two of the most popular techniques are the Branch and Bound method (Martin 1996) and Simulated Annealing (Yamada and Nakano 1996a). Other methods include Large Step Optimisation (Lourenço and

Zwijnenburg 1996), Genetic Local Search (Yamada and Nakano 1996b), the Shifting Bottleneck Procedure (Alvehus 1997) and Tabu Search (Nowicki and Smutnicki 1996). There are several other methods that are reviewed in the literature (Vaessens *et al*. 1996, Blażewicz *et al*. 1996).

However not many review papers have been published in the field of neural network based scheduling, which may be explained by the fact that until now only a few scheduling systems use neural networks. This paper adds to this area of knowledge by providing a detailed review of neural network based scheduling works and describing a scheduling system which uses a modified back-error propagation architecture.

The simplicity, along with their capability to perform distributed processing, as well as their ability to learn and generalise, have made neural networks a popular methodology, allowing them to be used in many real life applications (Zhang and Huang 1995). Recognising hand written letters, facial expressions, finding an optimum route for a travelling salesman problem (*TSP*) and scheduling a job-shop are just a few examples of problems that neural networks have successfully solved.

Cheung (1994) describes several neural network architectures such as searching network (Hopfield net), error correcting network (Multi - Layer Perceptron), probabilistic network (Simulated Annealing), competing network and self - organising network to solve the scheduling problem. Figure 1 shows their respective classifications within the field of neural computing.

Searching networks such as Hopfield nets are autoassociative non-linear networks, which have inherent dynamics to minimise the system energy function or Lyapunov function. The energy function is a non increasing vector which represents the activation in the units. The existence of such a function in association with a simple asynchronous updating rule enables the network to converge to a stable state of activations, rather than oscillating.

Changes in state during asynchronous updating always decrease the energy of the system monotonically, and the system dynamics always act to reduce the energy to a minimum. In general the energy function has many minima. As the system evolves with time, its energy decreases to one of the minimum values and maintains this value since no further energy changes are possible. These points are usually local minima.

Error correcting networks are trained on examples which take the form of a mapping $f: \mathcal{S} \subset \Re^n \rightarrow \Re^m$, from some arbitrary bounded subset $\mathcal{S}$ of n-dimensional Euclidean space to m-dimensional Euclidean space. When an activity pattern is applied to the network, the error correcting rule adjusts the synaptic weights in accordance with the above mapping. Specifically the actual response of the network is subtracted from the desired target response to produce the error signal. Weights are adjusted so the actual network response moves closer to the desired response.

Simulated Annealing is an iterative improvement scheme that has an analogy to statistical mechanics and combinatorial optimisation. Metropolis *et al.* (1953) introduced a simple Monte Carlo algorithm for simulating the transition of a liquid to a solid in such a way that thermal equilibrium is achieved. Kirkpatrick *et. al.* (1983) incorporated a time dependent temperature variant to the Metropolis algorithm, where at each temperature value the solid is allowed to slowly reach thermal equilibrium, characterised by a probability of the atoms being in the ground state, given by the Boltzmann distribution. This technique is applied to neurons, where their output is stochastically determined at each temperature a number of times. Then the temperature is progressively lowered and the process repeated until thermal equilibrium (optimisation) is achieved.

Competing and self organising networks are very similar, the only difference is that competing networks form internal clusters from binary values, while self organisation networks form clusters from continuous valued inputs.

In competitive learning the output neurons of the network compete among themselves to be activated or fired, with the result that only one output neuron, or one neuron per group (known as the winner-takes-all neuron) is activated at any one time. This neuron inhibits others in its neighbourhood from firing. A group of *M* cells within which such inhibitions occur is called an inhibitory cluster, and it has the property of decoding a distributed pattern into a one in *M* firing pattern. This is achieved by lateral inhibitory connections between the output neurons (Rosenblatt 1958).

In self organising networks the neurons are organised into local neighbourhoods that act as feature classifiers on the input data. During the self organisation process the cluster unit whose weight vector matches the input pattern most closely (the proximity is typically represented by the square of the minimum Euclidean distance) is chosen as the winner.

Π_G using neural networks has mainly been implemented in the first two paradigms, so the focus is on these two models only.

3. THE APPLICATION OF THE HOPFIELD MODEL TO Π_G

The Hopfield Network (Hopfield and Tank 1985) dominates neural network based scheduling systems (Foo and Takefuji 1988a, b, c; Zhou *et al.* 1991; Zhang *et al.* 1991; Chang and Nam 1993; Willems and Rooda 1994; Satake *et al.* 1994; Foo *et al.* 1995; Sabuncuoglu and Gurgun 1996) and is analogous to an electronic circuit consisting of operational amplifiers. As a result these models are simulated using such hardware components and as previously mentioned the dynamics are governed by an energy function, $\mathcal{E}$. When applied to Π_G, the aim is to minimise the energy function based on the makespan, $\mathcal{E}$ subject to various resource constraints, and if the constraints are violated, a penalty value is produced which increases the value of $\mathcal{E}$.

Foo and Takefuji (1988a, b, c) are the first to apply a neural network based system to Π_G. In their earliest work (Foo and Takefuji 1988a, b) they map Π_J onto a two dimensional

matrix mn by (mn+1) neurons. To avoid the problems of local convergence a simulated annealing (*SA*) process is then applied to the model. However there are several limitations in this method, the number of jobs must be greater than the number of machines; only small problems could be solved; there is no guarantee of an optimal solution; the *SA* method requires excessive computation. There is a large requirement for the number of neurons, $N=mn(mn+1)$, and even more interconnections, N^2, are required and hence it is concluded that a hardware implementation is not possible.

Foo and Takefuji (1988c) extend their previous work by formulating the scheduling problem as a set of integer linear equations, creating an Integer Linear Programming Neural Network (*ILPNN*). The energy function is represented by the sum of starting times of all jobs. This function is to be minimised while making sure the constraints are not violated, ie. all jobs are processed in a specified order, the right process sequence is followed and no machine processes two jobs simultaneously. By incorporating such a construction the requirement in the number of neurons and connections is reduced to $\frac{1}{2}[mn(n+1)]$ and $m^2n^3(n+1)$ respectively. However there is no guarantee of an optimal solution; only small problems are solved and the requirement for amplifiers and resistive connections increases as a high degree polynomial.

Zhou *et al.* (1991) propose a Linear Programming Neural Network (*LPNN*) to circumvent some of the shortfalls of the local convergence of the *ILPNN* used by Foo and Takefuji (1988c). They avoid the use of a quadratic energy function by implementing a linear function instead. This prevents the need for a conventional integer linear programming method which incorporates numerous control variables. By using this method the numbers of neurons and interconnections are drastically reduced, allowing problems of the size up to 20×20 to be tackled. However there is still no guarantee of an optimal solution for larger problems, the number of jobs must be equal to the number of machines and the system malfunctions if it is not applied to problems for which it is designed for.

Zhang *et al.* (1991) propose a similar method using a Hopfield network to that of Foo and Takefuji (1988c). They classify jobs into various categories of priority, depending on their importance. The energy equation is formulated in such a way it minimises the sum of the finishing times of all the jobs. An additional term is introduced in the energy equation that allows high priority jobs to be processed first, while the constraint equations prevent the early processing of low priority jobs. However this method is again limited to small problems and there is a tendency for the system to converge on non-optimal solutions.

Chang and Nam (1993) implement an *LPNN* which tries to converge on the global optimum. The system works very much on the same principle as that of Foo and Takefuji (1988c) except that it avoids the need for integer constraints by introducing slack variables. However only small problems are solved, a large number of constraints need to be formulated, numerous interconnections and neurons are required and there is still no guarantee of finding an optimal solution.

Willems and Rooda (1994) use an *ILPNN* where the search space is reduced by precalculation and the creation of minimum starting times, known as thresholds. Fast convergence to a feasible solution is promoted by the incorporation of feedback connections. However feedback can interfere with the evolution of an optimal solution. Hardware implementation of this system is difficult and it deals with small problems only.

Satake *et al.* (1994) simulate the Hopfield net on a digital framework incorporating a Boltzmann machine (Hinton *et al.* 1984), where the threshold values of the network are not predetermined but revised at each transition. By using this format the system is able to deal with larger problems than its predecessors. However the revision of each transition necessitates the need for a secondary energy equation which can produce non-feasible schedules.

The most recent attempt by Foo *et al.* (1995) to solve Π_G implements a modified Hopfield model, avoiding the use of a local optimum converging quadratic energy function by

the inclusion of non linear "*H*-amplifiers" into their circuit. This allows the "*G*-amplifiers" to be implemented as non-inverting amplifiers which obey a linear relationship. The output of the "*G*-amplifier" represents the starting time of each job while the "*H*-amplifier" output is a zero-one variable. By using a linear energy function the neurons are able to maintain their simple processing capabilities. However for Π_G, when m is fixed at 100 and n is varying, the number of constraints and variables required shows a polynomial growth, with the number of constraints approaching one million. In addition the number of amplifiers also exhibits a polynomial growth with more than 1.5 million, while the number of interconnections shows an exponential requirement with more than 1×10^{12} for a 100×100 instance. As these numbers are so astronomical hardware implementation is not feasible, no results are shown for large problems and also it appears there is no guarantee of finding an optimal solution.

The latest work on neural network based scheduling is by Sabuncuoglu and Gurgun (1996) who develop a modified 3D Hopfield model for the single machine mean tardiness problem and the minimum makespan job-shop scheduling problem (cf. Lo and Bavarian 1993). The approach differs from traditional Hopfield implementations in that an external processor is augmented to the network. Two operations requiring the same machine are randomly selected by the processor and swapped. The method of acceptance is analogous to a threshold acceptance algorithm with a 10 % threshold.

Experiments on 25 benchmark problems indicate that most problems are solved optimally. Although this is currently the best NN method for Π_G the method is highly problem dependent as it requires good initial values. Although the external processor acts as a metastrategy which guides the network, operations are randomly chosen for swapping. Balas (1969) proves that unless the operations are on the critical path swapping will never improve the makespan. Glover (1995) states that a random move selection is highly unproductive. Swapping can also result in non feasible solutions.

All the neurons need to be connected to the external processor suggesting the need for excessive numbers of connections which is in the order of mn^2. The technique only generates semi-active schedules therefore not guaranteeing optimality. There is a $O(mn^2)$ time requirement to compute the makespan. The termination criterion is inadequate because even if optimality is achieved the process continues searching and the computational requirement is very high. Even though hard problems such as LA 21, 27, 29, 38 (Lawrence 1984); ABZ 7, 8, 9 (Adams *et al*. 1988) are not attempted the authors have solved FT 10 (Fisher and Thompson 1963) which is a well known hard instance.

Until this point we have described systems based on the Hopfield network. However there are many scheduling systems which incorporate the back-error propagation algorithm.

4. THE APPLICATION OF *BEP* MODELS TO SCHEDULING PROBLEMS

There has been extensive research on neural computing (Lippmann 1987) and since the work of Rumelhart *et al*. (1986) on the multi-layer perceptron (*MLP*) supervised learning by the back-error propagation (*BEP*) algorithm has become the most popular method of training neural networks. Their implementation removed the major problem Perceptrons had in solving non-linear problems such as the exclusive-or (*XOR*), as pointed out by Minsky *et al.* (1969) and consequently the *BEP* paradigm has opened up a wide range of applications for neural networks (Gernoth and Clark 1995, Goodacre *et al.* 1995, Jadid and Fairbairn 1995, Pattichis *et al.* 1995, Wang and Teng 1995).

Rabelo and Alptekin (1989a, b; 1990a, b) provide one of the earliest studies in neural scheduling. Although this system is able to get lower tardiness values than six dispatch heuristics, the neural network does not perform any real scheduling but purely ranks priority rules and determines coefficients. The actual scheduling is done by the expert system module. Furthermore the neural models are not the main focus of this system but are only utilised in two sub modules. In addition the model only deals with small instances and considers merely a single performance criteria, that of tardiness.

Potvin *et al*. (1992) employ an 8-8-1 back-error propagation neural network for a dynamic vehicle dispatching problem (eg. ambulances, parcel delivery and taxis) where after 40,000 iterations the neural net has the same response as the expert for 89 % of the training examples and 78 % of the test examples. However as the model is only trained from the perspective of an expert, no guarantee of optimality is provided and the time to converge even using a *SPARC* workstation seems excessive. This method is fitting to be described as a "data retrieval" (look-up table) system. In a similar "data retrieval" situation the system developed by Jain (1995) produced a 98 % accuracy and also dealt with more complex problems.

Yih *et al*. (1991) propose a hybrid semi-Markov neural network method for scheduling a crane in a circuit board production line. Computational analysis indicates the hybrid system to be better than a human scheduler as throughput increased by 4 % and the error rate reduced from 3.90 % to 1.14 %. However the performance of the neural network is not reviewed, raising doubts about its relative success in training. There is also no discussion of the diversity of test and training data. The approach considers only a very simple and small problem $5/1/P/W_{max}$, which is among the most trivial class of problems to schedule and the neural network is not involved in the optimisation process but is only trained to look up the best policy.

Sastri and Malave (1991) apply two neural models, a Bayesian Classifier and a *BEP* network, to calculate the expected operation cost per period and the optimum control policy. Results achieved after comparable error convergence show that the Bayesian network has a 99.06 % policy identification accuracy and an overall mean error of 3.32 % in estimating the average operation cost per period while for the *BEP* model the corresponding results are 41.67 % and 16.93 % respectively. This suggests that real progress in back error propagation scheduling might be achieved by incorporating a proven mathematical optimisation model. Nevertheless the number of neurons required for this problem is large indicated by the fact that even for a system with 8 control policies and 3 operation cost states 75 neurons are required.

Hence for larger and more realistic instances network sizes will become excessive. The variation in training and test data is minimal which further emphasises the tendency of the model to be a "retrieval system." A 42 % accuracy in the *BEP* model suggests that the network is looking for generality between inputs and outputs, but is unable to find it.

Cedimoglu (1993) applies a *BEP* neural network approach to produce better sequences of jobs to be processed and results show that the neural model generally out performs several dispatch rules for various criteria. Nevertheless only smaller and simpler problems are considered; there is no guarantee of an optimal solution; the model acts as a schedule retrieval system producing a schedule for a given set of input parameters and as a result when new examples are presented the model struggles; there is no great divergence of training and test data and it is found that test data also contains training examples.

Keymeulen and De Gerlache (1993) propose a dual neural network for the rescheduling of an airline crew which is trained from the judgement of an expert. Two experiments are performed and the mean errors have been found to be $\cong 0.14$ and $\cong 0.1$. However the size of the network is not specified and there is doubt about the optimality of the training data since it is derived from an expert, which suggests that natural bias, human errors and inconsistencies could have been introduced. For test schedules that are not similar to training schedules convergence is not achieved, thus suggesting a deficiency in the system's capability to deal with generic problems.

Sim *et al.* (1994) claim that expert systems are not able to provide better results as they act basically as simple decision tables. A hybrid neural network and expert system (*NNSS*) simulation model is proposed to solve a dynamic Π_G and overcome this problem. Sixteen 14-14-1 *BEP* neural networks are embedded in an expert system. Each of the sub-networks correspond to an activation environment. The prevailing environment determined by the expert system, from ten scheduling factors, dictates which of the 16 networks is initialised. Each neural network is trained from the scheduling factors to recognise the individual contributions

of various heuristics in the activation environment subject to the criteria of average tardiness and percentage of late jobs. Results show the *NNSS* is able to match the best performance of each of the various dispatching rules and a composite rule expert system for 10 job lots of 5000 jobs.

However the method can be computationally demanding due to the time required to train 16 *BEP* networks and the time to run the training examples; only a small problem, nine machines, is dealt with; there is no evidence suggesting whether optimal solutions to these two criteria are obtained; given a set of scheduling factors, the neural network just determines a normalised value from which the expert system determines whether the job should be processed next, hence no optimisation is performed by the *BEP* model and there is no mention of the divergence of training and test data.

Kim *et al*. (1995) propose combining the back-error propagation paradigm with the Apparent Tardiness Cost (*ATC*) rule (Vepsalainen and Morton 1987). Here 90 samples, each sample containing 10 data sets, is used for training a 3-5-1 neural network. When the set up time is considered to be a separate entity then a 4-2-5-2 network is used with the Apparent Tardiness Cost with set-ups (*ATCS*) rule (Lee *et al*. 1992) to schedule the jobs. In this case 2250 samples, each with 10 data sets, are used for training. It is found that the system is able to successfully deal with problems from the training set as well as out with. Figure 2 gives a summary of how the above said models incorporate a back error propagation paradigm to scheduling problems.

Most of the problems that have been tackled by the *BEP* architecture are much smaller and simpler than Π_G; excessive numbers of neurons are required and training data is created from information specified by experts or from databases. Experts generally tend to get near optimal solutions in most domains. However because of the shear intractability of Π_G experts can only guess the solution which is anything but optimum. This does not satisfy present day

operating conditions where waste elimination is a primary concern, which demands, if not optimum, at least near optimum solutions.

This review so far indicates that Hopfield networks (Hopfield and Tank 1985) require excessive numbers of neurons and interconnections, hence they can deal with small problems only and do not guarantee optimal solutions as they are often trapped in local minima. On the other hand the traditional *BEP* models (Rumelhart *et al.* 1986) are generally applied to "easier" problems; require excessive numbers of neurons and are trained from non optimal data acquired either from an expert or a standard database. In many of these cases the optimisation is not performed by the neural network but instead left to an alternative technique.

A modified version of the *BEP* architecture is suggested to overcome many of the deficiencies of these two paradigms. Although as the traditional *BEP* model the modified *BEP* architecture also lacks a generalised capability the augmentation of additional properties allows transcendency of local minima. A novel input / output representation permits the number of neurons and interconnections to be minimal. The neural network performs the optimisation itself, thus not relying on alternative methods and the use of optimal training data ensures that the network is not just trained from mere "guesses".

5. THE APPLICATION OF A MODIFIED *BEP* MODEL TO Π_G

The modified *BEP* architecture incorporates the traditional forward and backward propagation with additional features such as a momentum parameter, η, a jogging parameter, β, and a learning rate parameter, α. These parameters alter the weight values, *W*, between neurons encouraging convergence away from local minima thus greatly improving the chances of solving Π_G successfully. The learning rate determines the amount of change in the weight values of the connections and usually reflects the rate of learning of the network. Values that are very large can lead to instability in the network and unsatisfactory learning, while values that are too small can lead to excessively slow learning. The momentum rate adds a fraction of the weight change to the next weight so as to prevent oscillations during learning (Rabelo and

Alptekin 1990a). This factor makes convergence of the network easier and faster (Lippmann 1987).

The symbols used to describe this algorithm are: t is a pattern, X is the target output, Q is the actual output, V is the neuron value, A, B and Z are the input, hidden and output layers respectively, Ψ is the error or difference between the actual and desired value, W is the weight, ΔW is the change in weight and θ is the bias or threshold.

In addition to the learning rate and momentum parameters, the choice of weights influences how quickly the network converges and whether it settles on a global or local optimum of the error function $\mathcal{E}$. In simple terms the i^{th} weight value between neurons in layer A and layer B is given by $W^i_{AB} = W^{(i-1)}_{AB} + \Delta W^i_{AB}$ where i = 1, 2, ..., n, ΔW^i_{AB} is the i^{th} weight change between these adjacent layers. $A \in \mathcal{R}$ and $B \in \mathcal{R}$, $\mathcal{R}$ is the set of all the layers in the network. $W^i_{AB} = \sum_{p=1}^{K}\sum_{q=1}^{L} w^i_{a_p b_q}$, and $\Delta W^i_{AB} = \sum_{p=1}^{K}\sum_{q=1}^{L} \delta w^i_{a_p b_q}$, $A = \{a_1, a_2, ..., a_K\}$ and $B = \{b_1, b_2, ..., b_L\}$, $w^i_{a_p b_q}$ is the i^{th} weight value and $\delta w^i_{a_p b_q}$ is the i^{th} weight change between the p^{th} neuron in layer A and the q^{th} neuron in layer B.[2] In the traditional *BEP* model, the weight adjustment of all neurons between two adjacent layers goes through a process of minimising the error function, culminating in a successful input - output representation. However, as shown in figure 3, there is a possibility that the n^{th} iteration weight value can become the same as the i^{th} iteration's.

The weight values therefore oscillate as they are unable to converge. The error cannot be further reduced as it is trapped in a local minimum, $\mathcal{E}^L*$. To avoid convergence towards $\mathcal{E}^L*$ the modified *BEP* model applies the jogging mechanism to the network.

[2] For completeness $W^i_{NET} = \sum_{J=1-2}^{Y-Z} W^i_J$ where the i^{th} weight value in the whole network (*NET*) is the sum of the i^{th} weight value between each layer. J = {1-2, 2-3, ..., *Y-Z*} the set of all adjacent layers in the network. This equation also corresponds to ΔW^i_{NET}.

Jogging involves a random alteration of the weights to find an alternative path to the desired output. W^j and β represent the jogged weight and randomly initialised jogging factor respectively. In figure 3, as $W_{AB}^{(n-1)}$ is converging on $\mathcal{E}^*$ the parameter β is initialised to encourage the weight values to approach $\mathcal{E}^*$, thus $W_{AB}^{j} = \beta W_{AB}^{(n-1)}$. This allows faster convergence towards a solution that is more likely to be a global optimum. In addition to jogging, extra neurons can be added into the hidden layer to encourage faster and improved convergence.

The likelihood of solving a problem can be greatly improved by finding an appropriate input representation, therefore the choice of input structure is vital. An input structure has to be chosen in such a way that the number of inputs grows linearly with the size of the problem. Robustness is also another important factor, the model must be able to handle raw input / output data without additional processing. Jain (1995) highlights that if the encoding of the Travelling Salesman Problem (*TSP*) is done in the form of intercity distances then ½[n(n-1)] input neurons are required, however the same information can be represented in coordinate form needing only 2n neurons. The assumption is that intercity distances are represented by the straight lines connecting them. Thus the requirement is reduced from a quadratic to a linear encoding which in a 1,000 city *TSP* is a reduction from 499,500 neurons to just 2,000 neurons.

A balance has to be made in such a way that the problem can be concisely encoded, yet the job sequence on machines should not be lost. The proposal here is to use a 2n input encoding for smaller problems which corresponds to the processing times of the jobs and the precedence order of machines. Due to their reduced dimensionality it is possible for all processing times required by $\mathcal{J}_i$ to be represented by a single input to neuron i where neuron (i+1) denotes the precedence order of $\mathcal{J}_i$. However for larger problems so as to maintain accurate problem representation, as all the processing times of a single job cannot be grouped

together, a 2*mn* encoding has to be applied where the inputs to the first *n* neurons represent the processing times of $\mathcal{J}_1$ on all the machines while the inputs to the next *n* neurons show the precedence order of machines required to process $\mathcal{J}_1$. This format continues for all the jobs and this data can be taken directly from the shop floor.

Another critical decision is the choice of output structure. The network's response should represent the optimum schedule. For smaller problems *n* outputs are used where each neuron corresponds to the job sequence on a machine. However for larger problems to maintain clarity *mn* outputs are adopted in order to represent job sequences explicitly for all the *m* machines, each machine has *n* outputs indicating the sequence of the jobs to be processed on that machine. The outputs from both of these formats can be implemented directly onto the shop floor.

The final decision concerns the number of hidden layers and hidden units. Pomerleau (1993) conducted experiments using either 0, 1 or 2 hidden layers[3] with between 0 and 70 hidden neurons in partially, as well as fully, connected structures. He concluded that there should be only one fully connected hidden layer containing a minimum number of neurons. The networks with additional hidden layers require significantly longer training times to reach the same level of performance. Furthermore networks with more than the minimum number of hidden neurons take longer to train because of the greater size of the network.

5.1 A scheduling system based on the modified *BEP* architecture

A system is described here which successfully schedules various sizes of Π_G ranging from 4×3 to 30×10. To provide a full understanding of the method used a smaller problem consisting of 6 jobs and 5 machines is described in detail and the results from a 30×10 problem are enumerated.

[3] No more than two hidden layers are chosen because Kolmogorov's theorem (Kolmogorov 1957) proves any problem can be represented by two hidden layers.

6×5 PROBLEM

The notation used in this paper to describe each instance will be that of Conway *et al.* (1967) where Π_G is represented by $A / B / C / D$. A is the number of jobs, B is the number of machines, C is the flow pattern within the machine shop and D is the performance measure by which the schedule is evaluated. Hence the problem here is notationally $6 / 5 / J / C_{max}$ where the performance measure C_{max} aims to minimise the makespan.

Although only a 6×5 instance is used in this example because of the exploding nature of the problem there are more than 1.9×10^{14} different possible schedules in comparison to only 14,000 in a 4×3 example. However industrial schedulers deal with thousands of parts and hundreds of machines which emphasise the difficulties they face. Table 1 specifies a $6 / 5 / J / C_{max}$ problem and a graphical solution in the form of a Gantt chart is given in table 2, where \ indicates that the machine is idle, waiting to process a job. The *SPT* (Shortest Processing Time) rule is used to schedule the jobs (Smith 1956) and the *FIFS* (First In First Served) rule (Rowe and Jackson 1956) is the tie breaker.

A two layer neural network, is used consisting of 12 inputs to represent processing times, and the process sequences of six jobs, and five outputs which encapsulate the sequences of jobs each machine needs to process. The process of learning in this network can be likened to a matrix transformation which maps a process sequence matrix into a job sequence matrix, as shown in figure 4.

The task here is to find the concise form of the data which caters for the reduced number of inputs and outputs, whilst maintaining the transparency, so that the neural net can find a plausible mapping. It should be possible to feed the processing times, in a raw form, into the network. However one needs to find a simple way of encoding the process sequence. Here the concept of branching trees, (figure 5), is used to relate every sequence to one of the branches and then each branch is assigned a number in ascending order.

Table 3 shows the training data obtained from the problem which is specified in table 1. In table 3 '13242' corresponds to the processing times of job 1 on the various machines (cf. table 1) while the second input '1' indicates the precedence order for this job. Based on the branching tree concept this second input '1' represents the precedence order of machines: ABCDE, '2' is equivalent to the precedence order ABCED, while '120' (which is $m!$) denotes the precedence order EDCBA. Such an encoding requires only $2n$ inputs.

Each output corresponds to the job sequence on every machine. Applying the branching tree approach the first output of '74' in table 3 is equivalent to a job sequence of $\mathcal{J}_1$, $\mathcal{J}_5$, $\mathcal{J}_2$, $\mathcal{J}_3$, $\mathcal{J}_6$, $\mathcal{J}_4$ which corresponds to row one of table 2. This scheme limits the number of outputs required to only n. When the example is used for testing, the output obtained is listed in table 4, to an accuracy of three decimal places. It can also be seen that the test output values are approaching the target values (compare table 4 with table 3). It is recognised that it may be impossible to get a perfect match. Table 5 denotes the training parameters used and table 6 shows the network parameters achieved from training.

The mean μ, and the standard deviation σ, of the optimum sequence of jobs produced by the neural network are compared with the target outputs used in training (table 7). This comparison is best achieved by two types of statistical tests known as a *t*-test and a *F*-test (Berenson *et al.* 1988). Both tests require the formulation of a set of hypothesis, H_0 and H_1. In the *t*-test, H_0 is accepted if no significant difference between the population means exists otherwise hypothesis H_1 is accepted. For H_0 to be accepted the t value calculated must lie within the specified hypothesis limits. The *F*-test is very similar to the *t*-test, but is a measure of the difference between population variances. Hypothesis H_0 is accepted when the F value calculated lies within the hypothesis limits, thus signifying the similarity between the two populations, otherwise it is rejected. The limits for both the *t*-test and *F*-test are set according to the size of the problem and can be found from statistical tables.

Learning is governed by a set of parameters (table 5). An upper limit, ε, is placed on the number of iterations. Once this limit is reached, training is stopped. Training can also be stopped once an acceptable difference, ϖ, between the neural network and target outputs, is achieved.

However learning cannot always be achieved to the desired ϖ value. If this is the case, then the modified *BEP* model is able to incorporate additional functions which improve learning. κ is the value at which the user specified parameters, λ, ν, ρ are triggered. If there is no decrease in ϖ after λ iterations then the weights are randomly altered. If after ν such alterations no improvement in ϖ is achieved then a neuron is inserted into the hidden layer. Once ρ neurons are inserted into the hidden layer, training is stopped.

From table 6 one could see that the network has undergone 25 iterations on 1920 good examples to achieve the required ϖ value. The training tolerance is a measure of the error when the process is stopped. Since learning is successful there are no additional neurons inserted into the network, so the network size stays the same. The training data for the 6×5 instances are created by generating a number of problem instances of this size and solving them by a simple dispatch rule. Hence the outputs are not optimal. Table 7 shows the means and variances of the results of the neural network from the simple dispatch rule solutions. As hypothesis H_0 is accepted for both the *t*-test and *F*-test, it indicates that training is successful as the outputs from the neural network have the same statistical parameters as that of the dispatch rule outputs.

30×10 PROBLEM

As the computational effort required for large instances of Π_G is excessive an exhaustive search method becomes impractical for the creation of training examples. Therefore benchmark instances proposed in the Π_G literature (Lawrence 1984, Adams *et al.* 1988, Taillard 1993) have been used as training data.

A two layer network is again used, with 300 nodes to input the processing times arising from 10 processes on every job and 300 input nodes to feed the precedence order for each job. While the 300 output nodes provide the optimum sequence of jobs to be processed by each machine. Here again it can be considered as a transformation which converts a matrix of process sequences for every job into a matrix of job sequences for every machine. The hidden layer has been arbitrarily chosen (Pomerleau 1993). As mentioned earlier for these 30×10 instances optimum training data is taken from the available literature. Table 5 describes the parameters used for training, with the results of training given in table 6, while table 7 provides a statistical analysis of the results of the neural network from the optimum solutions. One can conclude that training is successful, since the network is able to converge to a training tolerance of 0.020, there are no bad results and hypothesis H_0 is accepted for both tests.

5.2 Analysis of the modified *BEP* network

As indicated earlier because of the limited number of *BEP* based models for Π_G, providing a comparative study on the performance of the system is difficult. Vaessens *et al*. (1996) note that the application of neural networks to Π_G, mainly from the Hopfield domain, is at an initial stage and the reported results are minimal until now. Consequently neural network based scheduling systems are usually compared with traditional heuristics (Cherkassky and Zhou 1992, Smith *et al*. 1996). Cherkassky and Zhou (1992) compare the NN model of Zhou *et al*. (1991) with three priority dispatch rules: MWR, LWR and SPT on four problems taken from the Π_G literature and another 100 problems ranging from $n = 4$ to 10 and $m = 4$ to 10 which are randomly generated. Two performance measures are used, makespan and mean flow time minimisation. Results show that the neural network performs better in both criteria for all 104 problems except on one instance. However priority dispatch rules are known to give extremely poor solutions especially as the dimensionality increases.

Currently the modified *BEP* neural network system presented earlier can handle problems of size up to 30×10. Table 8 provides a comparative analysis of this method, *MBEP*, with three priority dispatch rules, SPT - shortest processing time; MWR - most work remaining and FCFS - first come first served, and the Shifting Bottleneck Procedure (SBP) of Adams *et al*. (1988). SBP is specifically chosen for the comparison as this was the first technique to solve these three large problems optimally. The results of these three simple dispatch rules have been performed on an IBM RS6000 by the authors. The *MBEP* system, which is run on a PC 486 with a clock speed of 33 MHz, gives comparable times to the priority dispatch rules while the makespans achieved are much shorter. When compared with SBP the makespans achieved are similar however the modified *BEP* model gets to the solution faster than SBP does. The superiority of this system and the inconsistent performance of priority dispatch rules is clearly brought out in table 9 where the total and average Relative Error and Computer Independent CPU times are given. The CI-CPU times are based on comparisons of Dongarra (1998) as interpreted by Vaessens *et al*. (1996).

6. DISCUSSIONS

6.1 Limitations in the various neural models presented

HOPFIELD

For complex problems such as Π_G the Hopfield model is unable to converge to the global optimum as it has a tendency to become trapped within local minima solutions, hence there is no guarantee of achieving good solutions. Many of these local solutions do not satisfy the specified constraints and hence produce sequences that violate precedence relationships or result in corresponding activities not being sequenced at all. The Hopfield based systems have difficulty in trying to solve large problems and the formulation of a quadratic energy function exhibits poor scaling properties as the number of amplifiers and resistive interconnections required increase exponentially and are excessive even for small problems. Kobayashi and Nohaka (1990) have shown that their knowledge based system provides better overall

schedules than the Hopfield network where the objective function is to minimise the daily resource usage.

In addition, the energy function relies on the assumption of symmetric connection weights but there is no evidence that such a structure exists in any biological memory system. This is commonly recognised as the main drawback of Hopfield auto-associative memory (Zhuang *et al.* 1996)[4]. The solutions obtained depend on the initial parameter values selected, where for each individual instance these values need to be carefully chosen, thus resulting in a technique which is highly dependent on the particular problem instance. The simulation of the differential equation has a complexity of $O(n^3)$ making it computationally expensive to obtain solutions for larger instances. Wilson and Pawley (1988); *DARPA* Study (1988); Aiyer *et al.* (1990) and Kunz (1991) highlight considerable discrepancies in this model and the Hopfield paradigm is considered to be inferior to known conventional heuristic algorithms for combinatorial optimisation, in terms of computational time and the quality of solution (Zhou *et al.* 1991).

THE BEP NEURAL MODEL

Although the *BEP* model is able to perform classification effectively it exhibits limited success in dealing with optimisation problems because of the inherent lack of generic patterns between inputs and outputs in optimisation problems. As a result when faced with scheduling problems the *BEP* architecture is insufficient. This is clearly observed from the earlier systems developed (Sastri and Malave 1991, Yih *et al.* 1991, Potvin *et al.* 1992, Keymeulen and De Gerlache 1993, Kim *et al.* 1995) which apparently have an error function such as makespan to be minimised, however the *BEP* architecture is unable to optimise the objective criteria. Hence alternative systems are developed which incorporate Operations Research techniques where

[4] These authors implement the concept of supporting functions, to replace the traditional energy function notion of Hopfield, and hence avoid the assumption of symmetric connection weights, making models more biologically plausible.

the neural network does not perform the optimisation but gives or takes values to or from other optimisation methods. The problems tackled by these systems are much smaller and simpler than Π_G; generic cases could not be solved and training data is created from the advice of experts or from databases which is insufficient in the context of Π_G.

MODIFIED BEP NEURAL MODEL

The limitation to 30×10 instances results from the restriction on the number of neurons and connections that can be used in the system. It is important to note that the times presented in table 8 are those once training is achieved where for some problems nearly 24 hours of learning is required to achieve satisfactory maximum output error values. To achieve these global minimum solutions optimal training data is required hence this method is only as good as the other methods available. Therefore this approach would not be able to solve open problems.

The system's primary deficiency is its limitation in dealing with examples out with the training set. Hence test examples are successful, subject to the condition they do not vary by more than 20 % from any training set example and they result in the same optimum makespan. In order to achieve this during training it is the processing times of non critical operations which are varied, by less than 20 %. Based on the work of Balas (1969), Matsuo *et al*. (1988) and Van Laarhooven *et al*. (1992) prove varying these operations will not affect the critical path and thus the overall makespan will remain the same. The essential thing is to ensure that by varying the processing time of a non critical operation, it does not become critical. To do this the slack (the difference between the current and latest finish time) of each operation is determined and the operations whose slack is greater than a 20 % change in their processing times are perturbed. Note such a perturbation has analogies to the problem space based search techniques of Storer *et al*. (1992, 1995).

However altering the problem variables slightly too much will result in the need to find alternative optimal training data and the requirement to redo the training process,

consummating in another lengthy learning phase. Therefore despite some success major weaknesses are clearly apparent in the modified *BEP* model consequently indicators point to using multiple tools which combine these approaches with an additional algorithm to solve the combinatorially exploding Π_G.

6.2 Future directions

The lack of real success in the modified *BEP* strategy encourages the consideration of hybrid systems where neural networks should be integrated into a practical engineering approach (Haykin 1994). Glover and Laguna (1993) also indicate that hybrids are emerging to bridge differences in their component technologies. The most fruitful solutions lie in the creation of hybrid methods that amalgamate neural networks with other approaches to solve Π_G.

The justification is that the incorporation of different methodologies balances out the deficiencies in individual approaches producing a more complete system to deal with the problem. The advantage of using hybrid models (Fu 1996) are :

- Systems which combine disparate technologies
- Systems which capitalise on the synergy of multiple technologies
- Systems which implement multiple levels or facets of activities from different perspectives

Metastrategies such as simulated annealing (SA); tabu search (TS); genetic algorithms (GAs) guide a local heuristic, embedded within their structure, through the search domain and hence are able to provide a superior method (cf. results of Vaessens *et al*. 1996). Therefore this creates a potential application where NNs can use their inherent parallel capabilities to search the solution space. As the NN normally converges to a local minima the metastrategy surmounts this solution by taking it away from the local minimum and then the NN is reapplied. Hence application of such methods to the Hopfield as well as *BEP* paradigms should avoid convergence to poor minima. Glover *et al*. (1995) discuss the tremendous scope available in combining TS to form hybrids models.

7. CONCLUSIONS

The extensive review highlights the fact that the traditional *BEP* model has the drawback of lacking a generalised learning capability to map inputs and outputs for intractable $\mathcal{NP}$-Hard problems, while the Hopfield model requires large numbers of neurons and interconnections. A modified *BEP* structure is suggested as it is able to deal with much larger and more complex problems than any previous method. If training is unsuccessful additional user defined parameters are incorporated to take the error function from a local into a global optimum. The model is trained on optimum data rather than relying on the guesses of experts. The robustness of the model allows data to be taken directly to and from the shop-floor without requiring additional filtering and modifications. Schedules can be retrieved very quickly for a known job-shop and the modified approach employs a novel input-output structure to encode large problems in such a way that the requirement of neurons grows linearly with problem size.

However test examples are successful only when they do not vary by more than 20 % from any training example. Although this requirement strikes at the heart of the generalisation capability of neural networks it should be noted that this is one of the first systems that has used a modified *BEP* model to solve Π_G using input-output mappings. No special effort is made to define a specific error function to minimise any parameters which are important in the scheduling context. On the other hand, a simple mapping of the input - output is made which, as expected, does not solve universal scheduling problems. Consequently to achieve a more generic capability we propose the formulation of hybrid technologies that combine the neural network's parallel processing proficiency to quickly locate local optima with a metastrategy such as tabu search to transcend poor minima.

ACKNOWLEDGEMENTS

The authors acknowledge the assistance provided by Professor Jim Hewit and Dr Paul Lewis of the Department of APEME, University of Dundee, Scotland in the preparation of

this paper and the anonymous referees for their detailed comments and their constructive criticisms which have greatly enhanced the quality of this paper.

REFERENCES

ADAMS, J., BALAS, E., and ZAWACK, D., 1988, The shifting bottleneck procedure for job-shop scheduling. *Management Science,* **34**(3), 391-401.

AIYER, S. V. B., NIRANJAN, M. and FALLSIDE, F., 1990, A theoretical investigation into the performance of the Hopfield model. *IEEE Trans. Neural Networks*, **1**, 204-215.

ALVEHUS, M., 1997, The shifting bottleneck procedure a survey. Graduate Course Report, Linköping University, Linköping, Sweden, March.

APPLEGATE, D., and COOK, W., 1991, A computational study of the job-shop scheduling problem. *ORSA Journal of Computing*, **3**, 149-156.

BALAS, E., 1969, Machine scheduling via disjunctive graphs: An implicit enumeration algorithm, *Operations Research*, **17**, 941-957.

BERENSON, M. L., LEVINE, D. M., and RINDSKOPF, D., 1988, *Applied Statistics: A First Course* (Englewood Cliffs, NJ: Prentice Hall).

BLAŻEWICZ, J., DOMSCHKE, W. AND PESCH, E., 1996, The job-shop scheduling problem: conventional and new solution techniques. *European Journal of Operational Research*, **93**(1), 23rd August, 1-33.

CARLIER, J., and PINSON, E., 1989, An algorithm for solving the job shop problem. *Management Science*, **35**(2), Feb., 164-176.

CEDIMOGLU, I. H., ,1993, Neural networks in shop floor scheduling. *Ph.D. Thesis*, School of Industrial and Manufacturing Science, Cranfield University, UK.

CHANG, S. H., and NAM, B. H., 1993, Linear programming neural networks for job-shop scheduling. *Proceedings of the IJCNN'93 International Joint Conference on Neural Networks*, Vol. 2, Nagoya, Japan, 25-29 Oct., pp. 1557-1560.

CHERKASSKY, V., and ZHOU, D. N., 1992, Comparison of conventional and neural network heuristics for job-shop scheduling. *Proceedings of the SPIE - International Society for Optical Engineering*, Orlando, Apr. 21-24, **2**, pp. 815-825.

CHEUNG, J. Y., 1994, Scheduling. In *Artificial Neural Networks for Intelligent Manufacturing*, edited by C. H. Dagli (London: Chapman and Hall), pp. 159-193.

CONWAY, R. W., MAXWELL, W. L., and MILLER, L. W., 1967, *Theory of Scheduling* (MA: Addison-Wesley).

COOK, S. A., 1971, The complexity of theorem proving procedures. *Proceedings of the Third Annual ACM Symposium on the Theory of Computing*, New York, pp. 151-158.

DARPA, 1988, *Neural Network Study* (Fairfax, Va.: AFCEA International Press), November.

DELL'AMICO, M., and TRUBIAN, M., 1993, Applying tabu - search to the job-shop scheduling problem. *Annals of Operations Research*, **4**, 231-252.

DONGARRA, J. J., 1998, Performance of various computers using standard linear equations software. Technical Report CS - 89 - 85, Computer Science Department, University of Tennessee, Knoxville, TN 37996-1301, Tennessee, USA, January 4.

FOO, S. Y., and TAKEFUJI, Y., 1988a, Stochastic neural networks for solving job-shop scheduling: Part 1. problem representation. In *Proceedings of the 1988 IEEE International Conference on Neural Networks*, edited by B. Kosko, San Diego, California, 24-27 July, Vol. 2, pp. 275-282.

FOO, S. Y., and TAKEFUJI, Y., 1988b, Stochastic neural networks for solving job-shop scheduling: Part 2. architecture and simulations. In *Proceedings of the 1988 IEEE International Conference on Neural Networks*, edited by B. Kosko, San Diego, California, 24-27 July, Vol. 2, pp. 283-290.

FOO, S. Y., and TAKEFUJI, Y., 1988c, Integer linear programming neural networks for job-shop scheduling. In *Proceedings of the 1988 IEEE International Conference on Neural Networks*, edited by B. Kosko, San Diego, California, 24-27 July, Vol. 2, pp. 341-348.

FOO, S. Y., TAKEFUJI, Y., and SZU, H., 1995, Scaling properties of neural networks for job-shop scheduling. *Neurocomputing*, **8**(1), 79-91.

FU, L. M., 1996, Knowledge and neural heuristics. *Tutorial 5 Notes IEEE ICNN'96 International Conference on Neural Networks*, 18:00 - 21:00, Sunday 2nd June, 106 pp.

GAREY, M. R., and JOHNSON, D. S., 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (San Francisco: W. H. Freeman).

GERNOTH, K. A., and CLARK, J. W., 1995, Neural networks that learn to predict probabilities - global - models of nuclear - stability and decay. *Neural Networks*, **8**(2), 291-311.

GLOVER, F., 1995, Tabu search fundamentals and uses, Working Paper, College of Business and Administration and Graduate School of Business Administration, University of Colorado, Boulder, Colorado, June (revised and expanded), 87 pp.

GOODACRE, R., TREW, S., WRIGLEY JONES, C., SAUNDERS, G., NEAL, M. J., PORTER, N., and KELL, D. B., 1995, Rapid and quantitative - analysis of metabolites in fermenter broths using pyrolysis mass-spectrometry with supervised learning - application to the screening of penicillium - chrysogenum fermentations for the overproduction of penicillins. *Analytica Chimica Acta*, **313**(1-2), 25-43.

GLOVER, F., and LAGUNA, M., 1993, Tabu Search, in *Modern Heuristic Techniques for Combinatorial Problems* (Oxford: Blackwell Scientific Publications), edited by C. R. Reeves, pp. 70-141.

GLOVER, F., KELLY, J. P., and LAGUNA, M., 1995, Genetic algorithms and tabu search: hybrids for optimization. *Computers and Operations Research*, Jan, **22**(1), 111-134.

HAYKIN, S., 1994, *Neural Networks A Comprehensive Foundation* (New York: M^{ac}Millan College Publishing).

HINTON, G. E., SEJNOWSKI, T. J., and ACKLEY, D. H., 1984, Boltzmann machines: constraint satisfaction networks that learn. Technical Report CMU-CS-84-119, Department of Computer Science, Carnegie Mellon University, Pennsylvania, USA.

HOPFIELD, J. J., and TANK, D. W., 1985, Neural computational of decisions in optimization problems. *Biological Cybernetics*, **52**, 141-52.

JADID, M. N., and FAIRBAIRN, D. R., 1995, Adjustments of error by neural networks for the shear-stress carried by the stirrups of a beam-column joint. *Expert Systems with Applications*, **9**(3), 257-269.

JAIN, A. S., 1995, Solving $\mathcal{NP}$-Hard combinatorially exploding problems using neural networks. *Honours Project Report*, Department of Applied Physics and Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland.

JOHNSON, S. M., 1954, Optimal two- and three-stage production schedules with set-up times included. *Nav. Res. Logist. Quart*., **1**, 61-68.

KEYMEULEN, D., and DE GERLACHE, M., 1993, Comparison training for a rescheduling problem in neural networks. In *Advances in Neural Information Processing Systems NIPS'6 Conference*, edited by J. D. Cowan, G. Tesauro, and J. Alspector, Denver, Colorado, Nov 29-Dec 2, (San Francisco: Morgan Kaufmann), pp. 801-808.

KIM, S. Y., LEE, Y. H., and AGNIHOTRI, D., 1995, A hybrid approach for sequencing jobs using heuristic rules and neural networks. *Production Planning and Control*, **6**(5), 445-454.

KIRKPATRICK, S., GELATT, C. D. Jr., and VECCHI, M. P., 1983, Optimization by simulated annealing. *Science*, **220**(4598), 13 May, 671-680.

KOBAYASHI, Y., and NOHAKA, H., 1990, Application of neural network to schedule integration in plant engineering, *International Neural Network Conference*, Paris, France, pp. 287-290.

KOHONEN, T., 1987, Representation of sensory information in self-organising feature maps, and relation of these maps to distributed memory networks. *SPIE 1987 The International Society of Optical Engineering*, Bellingham, Washington, March, pp. 248-259.

KOLMOGOROV, A. N., 1957, On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition (in Russian). *Doklady Akademiia Nauk SSR*, **114**(5), 953-956. (American Mathematical Society Translation, 1963, **28**, 55-59).

KUNZ, D., 1991, Suboptimal solutions obtained by the Hopfield-Tank neural network algorithm. *Biological Cybernetics*, **65**, 129-133.

LAWRENCE, S., 1984, Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. Supplement, Graduate School of Industrial Administration, Carnegie-Mellon University, Pennsylvania, USA.

LEE, Y. H., BHASKARAN, K., and PINEDO, M., 1992, A heuristic to minimise the total weighted tardiness with sequence dependent set-ups. Technical Report, IEOR Dept, Columbia University, New York, USA.

LIPPMANN, R. P., 1987, Comparison between neural nets and conventional classifiers. *Proceedings of the IEEE First International Conference on Neural Networks*, June, pp. 485-493.

LIPPMANN, R. P., 1987, An introduction to computing with neural nets. *IEEE ASSP Magazine*, April, 4-22.

LO, Z-P., and BAVARIAN, B., 1993, Multiple job-shop scheduling with artificial neural networks. *Comput. Electr. Eng.*, **19**(2), Mar, 87-101.

LOURENÇO, H. R. D., ZWIJNENBURG, M., 1996, Combining the Large-Step Optimization with Tabu-Search: Application to the job-shop scheduling problem. In *Meta-heuristics: Theory and Applications*, edited by I. H. Osman, and J. P. Kelly (Boston: Kluwer Academic Publishers), pp. 219-236.

MARTIN, P. D., 1996, A time-oriented approach to computing optimal schedules for the job-shop scheduling problem, *PhD. Thesis*, School of Operations Research & Industrial Engineering, Cornell University, Ithaca, New York 14853-3801, August.

MATSUO, H., SUH, C. J. and SULLIVAN, R. S., 1988, A controlled search simulated annealing method for the general job-shop scheduling problem. Working Paper #03-04-88, Graduate School of Business, The University of Texas at Austin, Austin, Texas, USA.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., and TELLER, E., 1953, Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, **21**(6), June, 1087-1092.

MINSKY, M., and PAPERT, S., 1969, *Perceptrons: An Introduction to Computational Geometry* (MA: MIT Press).

NOWICKI, E., and SMUTNICKI, C., 1996, A fast taboo search algorithm for the job-shop problem, *Management Science*, **42**(6), 797-813.

PATTICHIS, C. S., SCHIZAS, C. N., and MIDDLETON, L. T., 1995, Neural - network models in EMG diagnosis. *IEEE Transactions on Biomedical Engineering*, **42**(5), 486-496.

POMERLEAU, D. A., 1993, *Neural Network Perception for Mobile Robot Guidance* (MA: Kluwer).

POTVIN, J. Y., SHEN, Y., and ROUSSEAU, J. M., 1992, Neural networks for automated vehicle dispatching. *Computers Operation Research*, **19**(3/4), 267-276.

RABELO, L. C., and ALPTEKIN, S., 1989a, Using hybrid neural networks/expert systems for intelligent scheduling in flexible manufacturing systems, *IJCNN International Joint Conference on Neural Networks*, Washington, Jun. 18-22, vol 2, pp. 608.

RABELO, L. C., and ALPTEKIN, S., 1989b, Synergy of neural networks & expert systems for FMS scheduling. In *Proceedings of the Third ORSA / TIMS FMS: Operation Research Models and Applications*, edited by K. E. Stecke and R. Suri, Amsterdam, pp. 361-366.

RABELO, L. C., and ALPTEKIN, S., 1990a, Adaptive scheduling and control using artificial neural networks and expert systems for a hierarchical / distributed FMS architecture. *Proceedings of Rensselaer' sSecond International Conference on Computer Integrated Manufacturing, (IEEE)*, Troy, New York, May 21-23, pp. 538-545.

RABELO, L. C., and ALPTEKIN, S., 1990b, Synergy of artificial neural networks and knowledge-based expert systems for intelligent FMS scheduling. *IJCNN International Joint Conference on Neural Networks*, San Diego, CA, Jun. 17-21, vol 1, pp. 359-366.

ROSENBLATT, F., 1958, The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386-408. Reprinted in Anderson and Rosenfield, 1988, pp. 92-114.

ROWE, A. J., and JACKSON, J. R., 1956, Research problems in production routing and scheduling. *J. Indust. Engr.*, **7**, 116-121.

RUMELHART, D. E., HINTON, G. E., and WILLIAMS, R. J., 1986, Learning internal representations by error propagation. In *Parallel Distributed Processing*, edited by D. E. Rumelhart and J. L. M^{c}Clelland, vol. 1 (MA: MIT Press), pp. 318-362.

SABUNCUOGLU, I., and GURGUN, B., 1996, A neural network model for scheduling problems. *European Journal of Operations Research*, **93**(2), Sept., 288-299.

SASTRI, T., and MALAVE, C. O., 1991, Identification of the optimal control policy for the Markov decision process by back propagation. In *Intelligent Engineering Systems Through Artificial Neural Networks*, edited by C. H Dagli, S. R. T. Kumara, and Y. C. Shin (New York: ASME), pp. 873-881.

SATAKE, T., MORIKAWA, K., and NAKAMURA, N., 1994, Neural network approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics*, **33**(1-3), Jan., 67-74.

SHMOYS, D. B., STEIN, C., and WEIN, J., 1994, Improved approximation algorithms for shop scheduling problems. *SIAM J Comput*, **23**(3), June, 617-632.

SIM, S. K., YEO, K. T. and LEE, W. H., 1994, An expert neural network system for dynamic job-shop scheduling. *International Journal of Production Research*, **32**(8), Aug., 1759-1773.

SMIT, G. H., 1992, A hierarchical control architecture for job-shop manufacturing systems. *Dissertation*, Eindhoven University of Technology, Eindhoven, Netherlands.

SMITH, W. E., 1956, Various optimizers for single stage production, *Naval Research Logistics Quarterly*, **3**, 59-66.

SMITH, K. PALANISWAMI, M. and KRISHNAMOORTHY, M., 1996, Traditional heuristic versus Hopfield neural network approaches to a car sequencing problem. *European Journal of Operations Research*, **93**(2), Sept., 300-316.

STORER, R. H., WU, D. S. and VACCARI, R., 1992, New search spaces for sequencing problems with applications to job-shop scheduling. *Manage Sci*, **38**(10), 1495-1509.

STORER, R. H., WU, D. S. and VACCARI, R., 1995, Problem and heuristic space search strategies for job shop scheduling. *ORSA Journal on Computing*, **7**(4), Fall, 453-467.

TAILLARD, É., 1993, Benchmarks for basic scheduling problems. *European Journal of Operations Research*, **64**(2), 278-285.

VAESSENS, R. J. M., AARTS, E. H. L., and LENSTRA, J. K., 1996, Job-shop scheduling by local search. *INFORMS Journal on Computing*, **8**, 302-317.

VAN LAARHOOVEN, P. J. M., AARTS, E. H. L. and LENSTRA, J. K., 1992, Job shop scheduling by simulated annealing. *Operations Research*, Jan-Feb, **40**(1), 113-125.

VEPSALAINEN, A., and MORTON, T. E., 1987, Priority rules for job shops with weighted tardiness costs. *Management Science*, **33**, 1035-1047.

WANG, J., and TENG, T. L., 1995, Artificial neural - network - based seismic detector. *Bulletin of the Seismological Society of America*, **85**(1), 308-319.

WILLEMS, T. M., and ROODA, J. E., 1994, Neural networks for job-shop scheduling. *Control Eng. Practice*, **2**(1), Feb., 31-39.

WILSON, G. V., and PAWLEY, G. S., 1988, On the stability of the travelling salesman problem algorithm of Hopfield and Tank. *Biological Cybernetics*, **58**, 63-70.

YAMADA, T., and NAKANO, R., 1996a, Job-Shop scheduling by simulated annealing combined with deterministic local search. In *Meta-heuristics: Theory and Applications*, edited by I. H. Osman, and J. P. Kelly (Boston: Kluwer Academic Publishers), pp. 237-248.

YAMADA, T., and NAKANO, R., 1996b, A fusion of crossover and local Search. *ICIT'96 IEEE International Conference on Industrial Technology*, Shanghai, China, Dec 2-6.

YIH, Y., LIANG, T. -P., and MOSKOWITZ, H., 1991, A hybrid approach for crane scheduling problems. In *Intelligent Engineering Systems Through Artificial Neural Networks*, edited by C. H Dagli, S. R. T. Kumara, and Y. C. Shin (New York: ASME), pp. 867-872.

ZHANG, C. S., YAN, P. F., and CHANG, T., 1991, Solving job-shop scheduling problem with priority using neural network. *Proceedings of the IJCNN'91 International Joint Conference on Neural Networks*, Nov 18-21, Vol. 2, pp. 1361-1366.

ZHANG, H.-C., and HUANG, S. H., 1995, Applications of neural networks in manufacturing: A state-of-the-art survey. *International Journal of Production Research*, **33**(3), 705-728.

ZHOU, D. N., CHERKASSKY, V., BALDWIN, T. R., and OLSON D. E., 1991, A neural network approach to job-shop scheduling. *IEEE Transactions on Neural Network*, **2**(1), 175-179.

ZHUANG, X., SHI, H., and ZHAO, Y., 1996, A general auto-associative memory model, *ICNN'96 Proceedings of the IEEE International Conference on Neural Networks*, Washington DC, USA, June 2-6, pp. 549-554.

Job	*Machine Sequence and (Processing Times)*
1	A(1), B(3), C(2), D(4), E(2)
2	B(3), C(1), A(4), D(2), E(2)
3	E(4), D(3), A(1), C(2), B(1)
4	B(2), E(4), C(3), D(1), A(3)
5	D(5), A(2), E(1), B(3), C(4)
6	C(1), E(4), B(2), A(5), D(1)

Table 1. A 6×5 problem

TIME	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*
A	1	\	\	\	\	5	5	2	2	2	2	3	6	6	6	6	6	\	4	4	4
B	4	4	2	2	2	1	1	1	6	6	5	5	\	3	\	\	\	\	\	\	\
C	6	\	\	\	\	2	\	\	1	1	\	\	3	3	4	4	4	5	5	5	5
D	5	5	5	5	5	3	3	3	\	\	1	1	1	1	\	2	2	4	6	\	\
E	3	3	3	3	6	6	6	6	5	4	4	4	4	2	2	1	1	\	\	\	\

Table 2. Solution to the 6×5 problem on a Gantt chart using the *SPT* rule and *FIFS* rule

Example Number	1
Input	13242 1 31422 32 43121 116
	24313 46 52134 77 14251 69
Output	74 390 625 529 360

Table 3. Training data for the 6×5 problem

Example Number	1
Output	75.773 393.925 624.335
	529.703 366.396

Table 4. Outputs learnt by the neural network for the 6×5 problem

	Problem Description	
Training Parameters	6×5 problem	30×10 problem
Maximum Training Runs (ε)	999999	999999
Momentum Rate of Learning (η)	1.000	1.000
Learning Rate (α)	0.700	0.700
Maximum Output Error (ϖ)	0.050	0.007
Jog Coefficient (κ)	0.250	0.250
Iterations per jog action (λ)	50	50
Number of jogs before insertion (ν)	10	10
Maximum neurons to be inserted (ρ)	100	100

Table 5. Training parameters

	Problem Description	
Network Parameters	6×5 problem	30×10 problem
Iteration Number	25	109
Number of Good Results	1920	75
Number of Bad Results	0	0
Training Tolerance	0.050	0.020
Network Size at the Start	12-4-5	600-16-300
Network Size at the Finish	12-4-5	600-16-300

Table 6. Network parameters once training is successfully completed

	6×5 problem				30×10 problem		
	Target Output	Neural Network Output			Target Output	Neural Network Output	
μ	363.2000	364.0200		μ	4.5000	4.6281	
σ	186.7377	186.4305		σ	2.8723	2.8856	
	Value calculated	Hypothesis Limits	Hypothesis Accepted		Value calculated	Hypothesis Limits	Hypothesis Accepted
t-Test Values	-1.2128	± 2.8073	H_0	*t*-Test Values	-2.4038	± 2.5758	H_0
F-Test Values	1.1105×10^{-4}	0 to 7.88	H_0	*F*-Test Values	1.4757×10^{-1}	0 to 6.63	H_0

Table 7. Statistical comparison

		LA– Lawrence (1984)					ABZ - Adams, Balas and Zawack (1988)				
		Makespan Achieved					CPU Time (secs)				
Instance	Optimum	SPT	MWR	FCFS	ABZ'88	MBEP	SPT	MWR	FCFS	ABZ'88	MBEP
LA 31	1784	2284	1976	1875	1784	1784	0.01	0.01	0.01	38.3	0.01
LA 32	1850	2438	1901	2007	1850	1850	0.01	0.01	0.01	29.1	0.01
LA 33	1719	2314	1902	1841	1719	1719	0.01	0.01	0.01	25.6	0.01
SPT/MWR/FCFS – IBM RS6000				ABZ'88 – VAX 780/11					MBEP – PC 486 (33 Mhz)		

Table 8. Makespan and CPU times for the 30×10 instances LA 31, 32, 33

	Relative Error (%)					CI–CPU Time (secs)				
Instance	SPT	MWR	FCFS	ABZ'88	MBEP	SPT	MWR	FCFS	ABZ'88	MBEP
LA 31	28.03	10.76	5.10	0.00	0.00	0.19	0.19	0.19	4.596	0.009
LA 32	31.78	2.76	8.49	0.00	0.00	0.19	0.19	0.19	3.492	0.009
LA 33	34.61	10.65	7.10	0.00	0.00	0.19	0.19	0.19	3.072	0.009
∑	94.42	24.16	20.68	0.00	0.00	0.57	0.57	0.57	11.16	0.027
μ	31.47	8.05	6.89	0.00	0.00	0.19	0.19	0.19	3.72	0.009
TF	CI–CPU = TF × CPU					19	19	19	0.12	0.94

Table 9. MRE and CI-CPU times for LA 31, 32, 33

∑ = sum; μ = mean; TF = Transformation Factor
Relative Error = (UB - Optimum) / (Optimum)

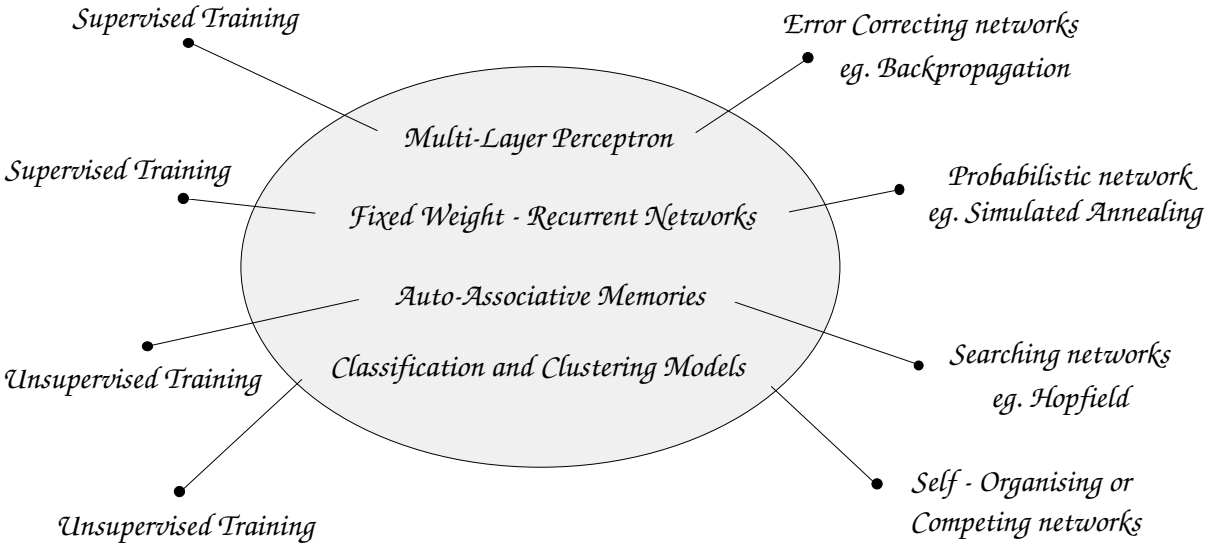

Figure 1. Classification of common neural network architectures

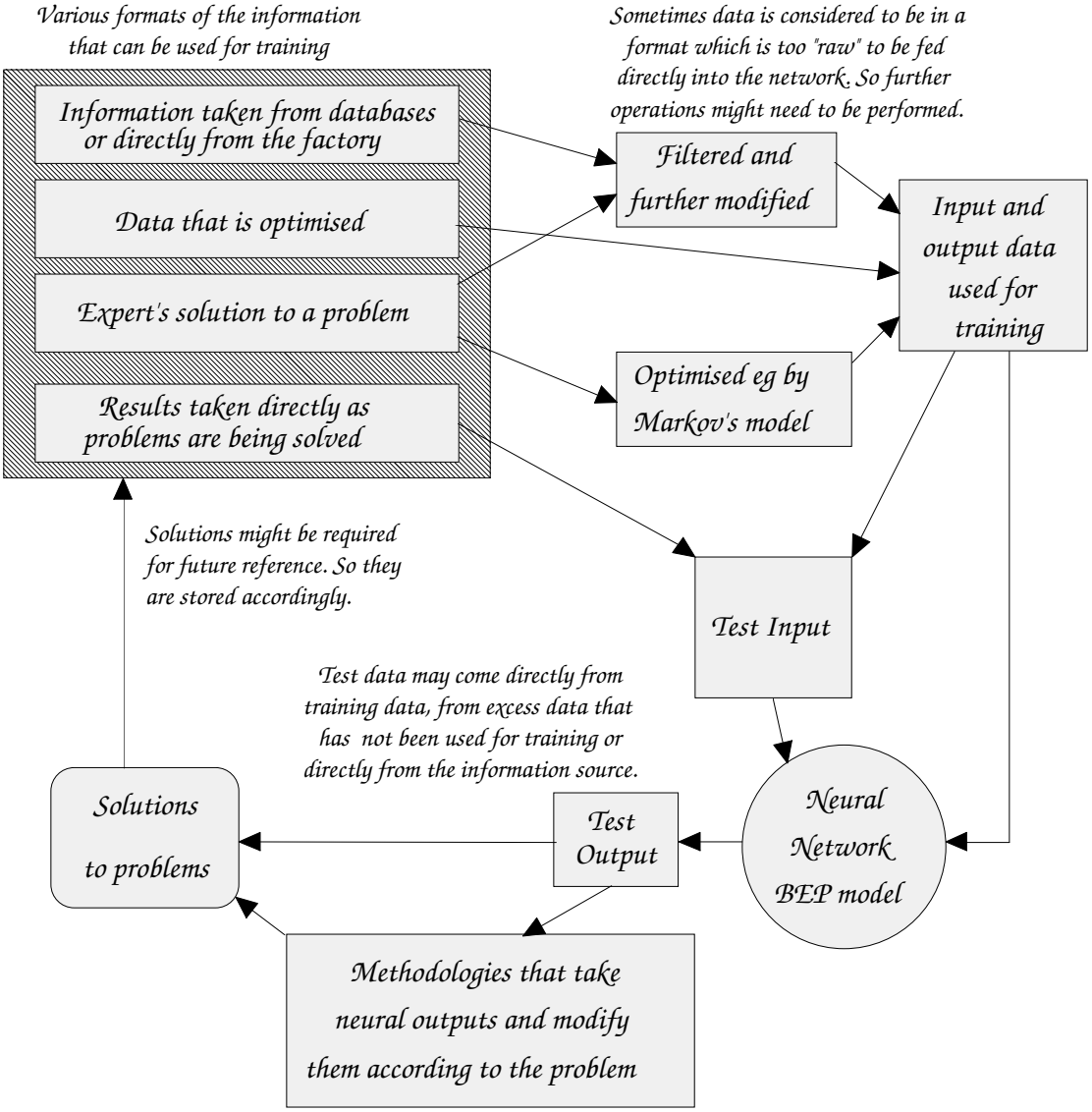

Figure 2. Back Error Propagation models in scheduling

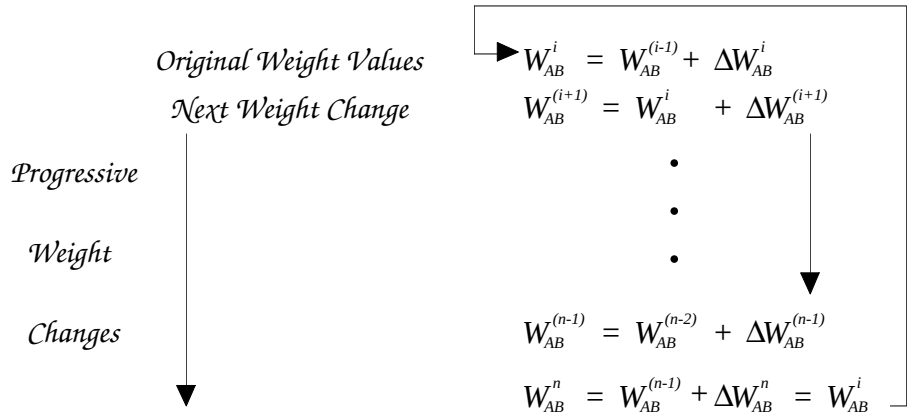

Figure 3. Weight convergence to a local minima

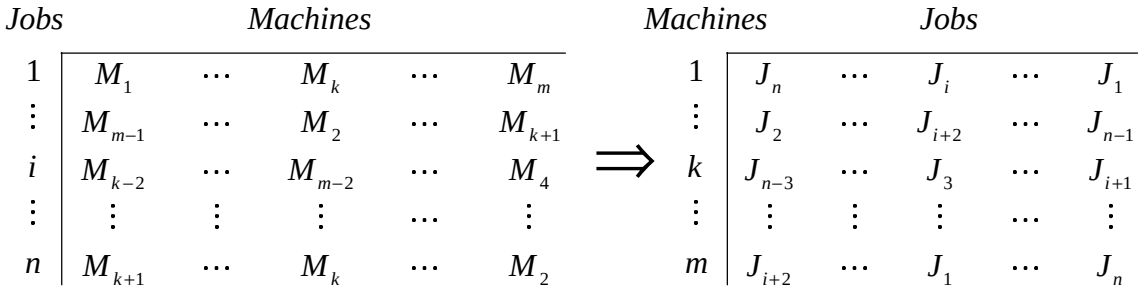

Figure 4. Conversion of process sequences to job sequences: A matrix representation

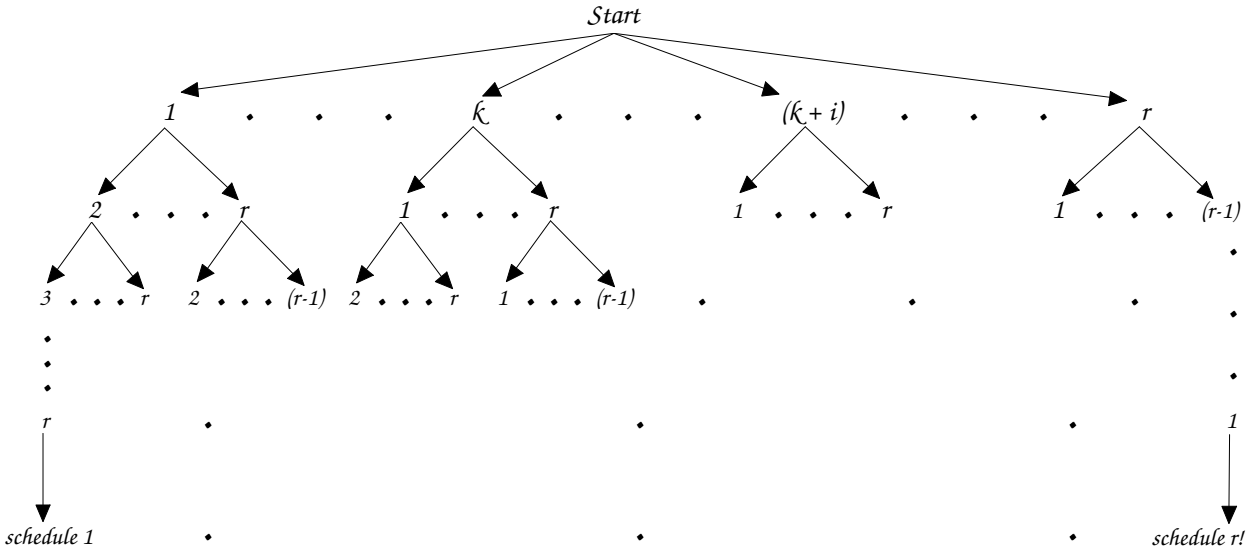

Figure 5. Unique identifications of sequences: A branching tree approach

LIST OF TABLES

LIST OF FIGURES