Neural networks for job-shop scheduling

Citation for published version (APA):

Willems, T. M., & Rooda, J. E. (1994). Neural networks for job-shop scheduling. *Control Engineering Practice*, 2(1), 31-39. [https://doi.org/10.1016/0967-0661\(94\)90571-1](https://doi.org/10.1016/0967-0661(94)90571-1)

DOI:

[10.1016/0967-0661\(94\)90571-1](https://doi.org/10.1016/0967-0661(94)90571-1)

Document status and date:

Published: 01/01/1994

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

NEURAL NETWORKS FOR JOB-SHOP SCHEDULING

T.M. Willems* and J.E. Rooda**

**Eindhoven University of Technology, Faculty of Mechanical Engineering, W-hoog, Room 1.126, P.O. Box 513,
5600 MB Eindhoven, The Netherlands*

***Eindhoven University of Technology, Faculty of Mechanical Engineering, W-hoog, Room 0.122, P.O. Box 513,
5600 MB Eindhoven, The Netherlands*

Abstract. A neural network structure has been developed which is capable of solving deterministic job-shop scheduling problems, part of the large class of np-complete problems. The problem was translated in an integer linear programming format which facilitated translation in an adequate neural network structure. Use of the presented structure eliminated the need for integer adjustments. Elementary precalculation is performed with the objective to reduce the search space allowing more rapid calculation of feasible solutions. In this precalculation the earliest possible starting times of the operations are calculated and set as thresholds in the network. The neural network structure was reliable in simulated operation and its performance was superior to structures which have been presented previously. The network structure always produces feasible solutions, in less time, without the application of integer adjustments.

Key Words. Job-shop scheduling, neural networks, integer linear programming, constraint satisfaction, optimization.

1. INTRODUCTION

A job-shop is a manufacturing facility which makes use of universal resources which can be used for many different manufacturing operations. Material to be processed flows through the job-shop by a variety of different routes, depending on the operations to be performed and the sequence in which they are to be performed. Scheduling in a job-shop is therefore a resource allocation problem which is subject to allocation and sequence constraints, an instance of the class of np-complete problems.

System engineers adopt two different approaches to scheduling: reactive and predictive scheduling. In reactive scheduling decisions are taken at the moment a decision point is reached. It is very probable that this will lead to a sub-optimal solution, since the scheduler does not fully anticipate future events. In predictive scheduling decisions are taken in advance. This results in an optimal schedule, but unforeseen system changes force a need for rescheduling, thus demanding a fast scheduling method. The problem with currently available constraint satisfaction techniques, in this regard, is that they suffer in general from combinatorial explosion, making them less suitable for np-complete problems.

A neural network is a functional abstraction of the biological neural structures of the central nervous

system. The simple neuron-like elements (units) that are massively connected to each other, are able to perform collective (parallel) processing. Neural networks can be trained to exhibit specific behaviour (Rumelhart and McClelland, 1986), and they can also be predefined to perform constraint satisfaction / optimisation tasks (Hopfield and Tank, 1985). It has been shown (Foo and Takefuij, 1988a-c; van Hulle, 1991) that constraint satisfaction and optimisation neural networks have the potential to solve the predictive job-shop scheduling problem in a satisfactorily rapid manner, but the neural networks presented in the literature all suffer, to a greater or lesser degree, from design deficiencies. If they are applied to problems other than those for which they were designed, they malfunction. Furthermore, some of them require an unrealistic amount of information so that they can be fine-tuned.

In this paper, a neural network structure is presented onto which an integer linear representation of a job-shop scheduling problem can be mapped. Simulation has shown that the network finds optimum and near-optimum solutions. The proposed structure is superior in the design of the feedback connections. The proposed structure guarantees feasible solutions. Due to the selected simulation approach, convergence is deterministic. Randomness can be introduced by simulating processing of the units in random order.

The search-space is truncated by precalculation and the setting of minimum starting times (thresholds). This affects the calculation in both a positive and a negative manner: solutions are found more rapidly, but the threshold may interfere with the evolution to an optimum solution. An additional layer of units eliminates the need for integer adjustments during calculation. This results in more rapid calculation of a schedule. The selected activation function of the units representing the calculated starting times prohibits the calculation of negative (impossible) starting times.

This paper is organised as follows: after a short introduction in neural network technology, the job-shop scheduling problem is introduced and explained, together with an integer linear representation of the problem. A neural network structure capable of solving the scheduling problem is presented. The problem-specific integer linear equations can be mapped on this neural network structure, resulting in a problem specific neural network.

The neural network structure is applied to a small job-shop scheduling problem, which is simulated, together with a more-complex problem. The small (2/3/J/Cmax) problem was solved after 42 cycles with precalculation and 158 cycles without precalculation. A larger (4/3/J/Cmax) problem was solved in an average of 142 cycles with precalculation and 224 cycles without precalculation.

2. NEURAL NETWORKS

Neural networks may be viewed as a collection of communicating simple processing elements. These elements are a functional abstraction of the neurones in the central nervous system. Here the elements are called units, rather than 'artificial neurones' to avoid the suggestion that any biological plausibility is claimed.

A unit is a simple processing element, connected to other units by its 'weighted' (dendritic) connections, see Fig. 1.

Fig. 1. Unit

A unit collects weighted (W_1 to W_n) numerical information from other units (A_1 to A_n). This information, sometimes increased with a bias, is summed resulting in the net input (N_i). The N_i is passed through an activation function F , resulting in the activation (A_i) of the unit.

The type of activation function implemented in a unit determines the functionality of the unit. The activation functions used in the proposed network structure are described in Section 5. After the summed input has been passed through the activation function, the activation (A_i) is collected by other units that are connected to this unit. The activation function may be (non)deterministic binary or (non)deterministic continuous. The activation function is selected according to the functionality required of the neural network.

Neural networks can perform two basic functions: they can be trained to remember some information (Rummelhart and McClelland, 1986), and they can be used to perform constraint satisfaction and optimisation tasks (Hopfield and Tank, 1985; Tank and Hopfield, 1986; Poliac *et al.*, 1987). The job-shop scheduling problem will be tackled by the latter type.

Hopfield and Tank (1985) introduced a deterministic neural network model capable of solving constraint satisfaction and optimisation problems by translating the problem in a number of units with predefined fixed weighted connections. Examples of problems that have been solved in this way are the 'eight queens problem' and the 'travelling salesman problem' (Hopfield and Tank, 1985), crew scheduling (Poliac *et al.*, 1987) and others.

3. JOB-SHOP SCHEDULING

A manufacturing system consists of a collection of resources on which operations are to be performed. There are several basic resource layouts (Smit, 1992) of which the job-shop is the most flexible one. The job-shop contains universal resources combined with a great route flexibility. The jobs that arrive at a job-shop consist of a prescribed, rigid sequence of operations (recipe). Which resource an operation has to be performed on, the allocation decision, is determined by the scheduler. The order in which the different operations of the available jobs have to be performed by the resources, the sequence decision, is also determined by the scheduler. The scheduler has to create a schedule that fulfils a specified performance criterion, resulting in the optimisation of the defined manufacturing system objectives. Different manufacturing systems with different objectives, or even different products, require different scheduling criteria. Several performance measures such as stocksize, maximum throughput, due date reliability and mean lead time, are accepted as the main scheduling performance criteria. Several other criteria can be derived from these criteria, such as minimisation of the makespan, minimisation of the maximum lateness, or minimisation of the summed lateness. Although the machine utilisation degree remains widely accepted as a criterion, it appears to be only the resultant of an optimisation criterion. Which measures or derivatives are used as a performance criterion of the scheduler, depend on the manufacturer's intentions.

In the literature, minimisation of the makespan is often used as the criterion, and it will also be used as criterion in this paper. The search for an optimum schedule is bound by two types of constraints. The first constraint states that the compulsory operation sequence (recipe) is to be guaranteed; the second constraint states that not more than one job can be processed on one resource at the same time. Scheduling can be viewed as optimisation, bound by sequence and resource constraints. A general definition of scheduling is: *scheduling is the allocation of resources over time to perform a collection of tasks* (Baker, 1974).

A more abstract version of the job-shop scheduling problem is mostly encountered in the literature (e.g. Foo and Takefuji, 1988a-c; Van Hulle, 1991; Zhou *et al.*, 1990). The general job-shop scheduling problem, as it has been called (French, 1982), is deterministic and no allocation decisions need to be taken. This means that for most of the job-shop scheduling problems investigated, only sequencing decisions have to be taken, operation times are deterministic, and no machine failure is encountered. An example of a general job-shop scheduling problem is a job-shop with three machines that have to operate on two jobs, all operating times and allocations being fixed. The general job-shop scheduling problem will be used in this paper to exemplify the proposed neural network.

Job-shop scheduling belongs to the class of np-complete problems. This means that the calculation effort required to find a solution to a job-shop scheduling problems grows exponentially or faster with the growth of the problem size. The class of np-complete problems contains many other problems, like the scheduling of operators to machines, managers to departments, instructors to courses, etcetera. Job-shop scheduling has been extensively studied partly because it is an instance of the np-complete class of problems.

Several notation systems have been presented for the representation of a specific job-shop scheduling problem, with its criterion. In this paper the notation system of Conway *et al.* (1967) will be used. This is a convenient way of presenting a specific scheduling problem in which:

n = number of jobs

m = number of machines

A = operation pattern (e.g. J = Job-shop)

B = performance criterion (e.g. Cmax = minimisation of the maximum completion time, equalling the minimisation of the makespan).

A two-job, three-machine, job-shop scheduling problem with minimisation of the makespan as performance criterion is represented as: 2/3/J/Cmax.

4. INTEGER LINEAR PROGRAMMING REPRESENTATION

Due to the difficulties encountered in solving job-shop scheduling problems, several representational aids have been developed to facilitate schedule generation (Baker, 1974). Graphs and Gantt charts, for instance, are used by human schedulers. In this article the Gantt chart will only be used to represent the results generated; an integer programming approach will be used to represent the constraints (Baker, 1974).

The constraints, a predefined operation sequence (sequence constraints) and the constraint that it is not allowed to operate more than one job on one resource the same time (resource constraints), can be translated to integer linear programming format in the following manner. The triplet ijk will be used to represent the operation j of job i on machine k . For each ijk a processing time t_{ijk} is known, and after the scheduling for each ijk a starting time is determined, denoted by S_{ijk} . The objective of job-shop scheduling is to find a set of starting times (S_{ijk}) which comply with the constraints, minimising the objective criterion.

To represent the constraints of a job-shop scheduling problem in an integer linear format, the sequence constraints have to be represented first. Suppose that the recipe of a job i states that operation j of job i requiring machine l is to be performed after operation $(j-1)$ of job i requiring machine k is finished. Then, for a feasible schedule, it is necessary that $S_{ijl} \geq S_{i(j-1)k} + t_{i(j-1)k}$, which can be reformulated as

$$S_{ijl} - S_{i(j-1)k} - t_{i(j-1)k} \geq 0. \quad (1)$$

There are $n(m-1)$ sequence equations for a $n/m/J/B$ job-shop scheduling problem.

The general resource constraint is of a disjunctive type that can be represented by two equations. If job i precedes job p on machine k then operation ijk must be finished before job p can be processed on machine k . $S_{pjk} \geq S_{ijk} + t_{ijk}$, which can be reformulated as

$$S_{pjk} - S_{ijk} - t_{ijk} \geq 0. \quad (2)$$

If, on the other hand, job p precedes job i on machine k then $S_{ijk} \geq S_{pjk} + t_{pjk}$, which can be reformulated as

$$S_{ijk} - S_{pjk} - t_{pjk} \geq 0. \quad (3)$$

Only one of these two statements is valid (disjunctive statements), depending on the (partial) schedule generated. To accommodate these two constraints in the representation, an indicator variable, Y_{ipk} , is formulated.

This variable indicates which statement is valid, i.e. it states whether job i precedes job p on machine k (value 1) or not (value 0). Thus the resource constraints become:

$$S_{pjk} - S_{ijk} + H(1 - Y_{ipk}) - t_{ijk} \geq 0 \quad (4)$$

$$S_{ijk} - S_{pjk} + H * Y_{ipk} - t_{pjk} \geq 0 \quad (5)$$

$$\text{with: } Y_{ipk} = 1 \text{ if } S_{ijk} \leq S_{pjk}$$

$$Y_{ipk} = 0 \text{ if } S_{ijk} > S_{pjk}$$

The constant H should be large enough to ensure that one of the disjunctive statements holds while at the same time the other statement is eliminated due to H . To fulfil this requirement, H should be larger than the largest possible starting time difference between two operations. This value can be calculated by adding all processing times (worst possible schedule):

$$H > \sum_{i=1}^n \sum_{j=1}^m t_{ijk}. \quad (6)$$

There are $nm(n-1)$ resource constraint equations for a job-shop scheduling problem, resulting in a total of $n(mn-1)$ equations for a job-shop scheduling problem. These general integer linear equations representing the constraints will be used for the translation of specific job-shop scheduling problems to a format that can be mapped on a neural network.

5. DESIGNING A NEURAL NETWORK FOR THE JOB-SHOP SCHEDULING PROBLEM

The integer linear representation described above has to be translated to a neural network structure that is capable of solving the problem. A general neural network structure is proposed, on which each specific job-shop scheduling problem can be mapped.

5.1 Units

The job-shop scheduling neural network should contain units that are capable of representing the starting times of the operations (S units), whether sequence constraints are violated (SC units), whether resource constraints are violated (RC units), and the value of the Y_{ipk} indicator variables (Y units).

S units. The input N_i of these units is calculated by adding the previous activation $A_{i(t-1)}$ (thus simulating a capacitor) to the summed incoming weighted activation.

$$N_i = \sum_{j=1}^n (W_{ij} \cdot A_j) + A_{i(t-1)} \quad (7)$$

The selected unit for this is of a deterministic linear threshold type with an activation A_i , a threshold x and a net input N_i .

$$A_i = \begin{cases} x & N_i \leq x \\ N_i & N_i > x \end{cases} \quad (8)$$

Since an operation can never start before time 0, the starting time of the entire schedule, the threshold x can be set at 0, resulting in the implementation of a non-negativity constraint.

The thresholds can also be determined in a problem-specific manner by calculation of the earliest possible starting time of the operations. The earliest possible starting time of the first operation of a job is the starting time of the scheduled time-span: 0 in the example. The second operation's ($i(j+1)k$) earliest possible starting time is $0 + t_{ijk}$. The earliest possible starting time of the third operation of job 1 is $0 + t_{ijk} + t_{i(j+1)k}$. In this manner the thresholds of the units representing the starting time are determined, thus reducing the 'search space' and resulting in a more-stable network.

SC and RC units. The net input of these units is calculated by adding the bias to the summed incoming weighted activation:

$$N_i = \sum_{j=1}^n (W_{ij} \cdot A_j) + B_i \quad (9)$$

The bias (B_i) added to the incoming weighted activations of the connected units is the processing time of the operation as formulated in the equation this unit represents. The constraint representing units are of a deterministic negated linear threshold type with x being the threshold and N_i the input:

$$A_i = \begin{cases} x & N_i \geq x \\ -N_i & N_i < x \end{cases} \quad (10)$$

This activation function allows for an indication of the violation of the equation being represented. The bias is applied to represent the problem-specific operation processing times.

Y units. The net input of the Y units is calculated by summation of the incoming weighted activation:

$$N_i = \sum_{j=1}^n (W_{ij} \cdot A_j) \quad (11)$$

The activation is determined according to a deterministic step function:

$$A_i = \begin{cases} 1 & N_i \leq x \\ 0 & N_i > x \end{cases} \quad (12)$$

The activation of this unit represents whether job 1 precedes job 2 on machine k or job 2 precedes job 1.

The proposed structure consists of three layers; the bottom layer containing the S units, the middle layer containing the SC and RC units, and the top layer containing the Y units. To fully represent the constraints, weighted connections between units have to be placed.

5.2 Connections

First the connections to the SC units will be described.

The constraint equation: $S_{ijb} - S_{i(j-1)a} - t_{i(j-1)a} \geq 0$ is represented by an SC unit. The SC unit collects the activation (representing the S_{ijk}) from the adequate S units of the bottom layer. To do so correctly, the S unit representing the starting time S_{ijb} should be connected with a weight of +1 and the S unit representing $S_{i(j-1)a}$ with a weight of -1. See Fig. 2. The bias of the SC unit consists of the negated $t_{i(j-1)a}$. The received input, together with the bias of the SC unit, indicates whether the represented constraint is violated by the suggested starting times. If the constraint is violated, the activation of the unit becomes greater than zero. This activation should be applied as a corrective signal for the S units; S_{ijb} should be increased and $S_{i(j-1)a}$ should be decreased, resulting in a positive weighted (e.g. +0.1) feedback connection to S_{ijb} and a negative weighted (e.g. -0.1) feedback connection to $S_{i(j-1)a}$. See Fig. 2.

Fig. 2. General structure for sequence constraints

For the implementation of the resource constraints in a general unit type as described above, the value of H should be implemented in the bias of the RC unit.

This requires a reformulation of the H containing part of the constraint equation:

$$H(1 - Y_{ipk}) = (-H * Y_{ipk}) + H,$$

allowing the implementation of H as a bias and a weight value of the connection to the Y_{ipk} unit.

$$\begin{aligned} S_{pjk} - S_{ijk} + H(1 - Y_{ipk}) - t_{ijk} &\geq 0 \\ \rightarrow S_{pjk} - S_{ijk} + (-H * Y_{ijk}) + \underline{H - t_{ijk}} &\geq 0 \end{aligned} \quad (13)$$

$$\begin{aligned} S_{ijk} - S_{pjk} + H * Y_{ipk} - t_{pjk} &\geq 0 \\ \rightarrow S_{ijk} - S_{pjk} + (+H * Y_{ijk}) - \underline{t_{pjk}} &\geq 0 \end{aligned} \quad (14)$$

In this way the RC equation can be implemented in a general RC unit with the underlined part

implemented as a bias, and H as a positive or negative weight value of the connection to Y_{ipk} . The Y units should receive information from the S units such that S_{ijk} preceding S_{pjk} results in an activation of 1. Therefore the S_{ijk} representing unit should be connected with a negative (-1) weighted connection and the S_{pjk} representing unit with a positive (+1) weighted connection. See Fig. 3.

Fig. 3. General structure for resource constraints

The RC units should receive information from the Y units as prescribed by the equations resulting in a connection weight of $-H$ for the RC unit representing the first equation (13) and of $+H$ for the RC unit representing the second (14) equation. See Fig. 3. Furthermore the RC units have to receive information from the S units. The RC unit representing the first equation should have a positive (+1) connection weight to the S_{pjk} representing S unit and a negative (-1) weighted connection to the S_{ijk} representing S unit. The RC unit representing the second equation should have a negative (-1) connection weight to the S_{pjk} representing S unit and a positive (+1) weighted connection to the S_{ijk} representing S unit. This information, together with the underlined part of the equation as a bias of the respective RC units, suffices to determine whether the proposed starting times violate the represented constraint.

A violated resource constraint should result in a modification of the starting times of the responsible S units. To achieve this, the S units collect the activation of the RC units through weighted connections to the RC units. The sign of the weights of these feedback connections cannot be unequivocally prescribed. If two operations are scheduled on one resource at the same time, it is not easily determined which operation should be advanced and which operation should be delayed. Here some rules of thumb like 'shortest processing time first' can be implemented by the correct setting of the weights. Small weights of these connections diminish the chance of getting stuck in a local minimum, at the cost of longer processing times.

5.3 Example

This general neural network structure will be explained with the use of a small example.

As an example a 2/3/J/C_{max} scheduling problem is used with its machine allocations and operation times presented in Tables 1 and 2.

Table 1.
Machine assignments

	Operation					
	1	2	3	1	2	3
Job	1	1	2	3	2	3
	2	3	1	2		

Table 2.
Processing times

	Operation					
	1	2	3	1	2	3
Job	1	5	8	2		
	2	7	3	9		

Before a dedicated neural network can be designed an integer linear programming representation has to be created according to the method presented. For the sequence constraints this translation results in:
 $n(m-1) = 4$ sequence constraints of type

$$S_{ijb} - S_{i(j-1)a} - t_{i(j-1)a} \geq 0,$$

- 1) $S_{122} - S_{111} - 5 \geq 0$
- 2) $S_{133} - S_{122} - 8 \geq 0$
- 3) $S_{221} - S_{213} - 7 \geq 0$
- 4) $S_{232} - S_{221} - 3 \geq 0$.

For the resource constraints the value of the constant H must be determined. This constant should have a value that is large enough to ensure that one of the disjunctive statements holds and the other is eliminated so:

$$H > \sum_{i=1}^n \sum_{j=1}^m t_{ijk} \quad H > 34, \quad H = 35.$$

For the example problem there are $nm(n-1) = 6$ resource constraints of type

$$S_{pjk} - S_{ijk} + (-H * Y_{ijk}) + \frac{H - t_{ijk}}{2} \geq 0$$

$$S_{ijk} - S_{pjk} + (+H * Y_{ijk}) - \frac{t_{ijk}}{2} \geq 0,$$

- 5) $S_{221} - S_{111} + (-35 * Y_{121}) + 35 - 5 \geq 0$
- 6) $S_{111} - S_{221} + (35 * Y_{121}) - 3 \geq 0$
- 7) $S_{232} - S_{122} + (-35 * Y_{122}) + 35 - 8 \geq 0$
- 8) $S_{122} - S_{232} + (35 * Y_{122}) - 9 \geq 0$
- 9) $S_{213} - S_{133} + (-35 * Y_{123}) + 35 - 2 \geq 0$
- 10) $S_{133} - S_{213} + (35 * Y_{123}) - 7 \geq 0$.

In total there are $n(nm-1) = 10$ equations, $nm = 6$ starting time (S_{ijk}) variables and $mn(n-1)/2 = 3$ disjunction (Y_{ipk}) variables for the 2-job, 3-machine problem.

The first layer of the neural network designed for the example problem consists of 6 S_{ijk} units, the second layer consists of 10 constraint units and the third layer consists of 3 Y units. See Fig. 4.

The thresholds of the S units can be determined in a problem-specific manner. For instance the threshold of the S unit representing the third operation of job 1 (S_{133}) can be set at 13 ($t_{111} + t_{122}$).

First the example will be applied to clarify the functionality of the general neural network structure for the sequence constraints. The sequence constraints are implemented in the general structure presented in Fig. 2. The problem-specific elaboration of this general structure is presented in Fig. 4.

The first unit of the second layer (representing the first sequence equation) for instance, collects negated information (connection weight -1) from the unit representing S_{111} and positive information (weight +1) from the unit representing S_{122} . Together with the bias of this unit (-5) the violation of this constraint can be determined. If, for instance, $S_{111} = 1$ and $S_{122} = 2$, a constraint violation should be signalled since the second operation starts before the first operation has ended. The net input of the first constraint unit, SC1 in Fig. 4, will be $2 - 1 - 5 = -4$, resulting in an activation of 4, signalling a violation. This information has to be fed back to the S units to cause an appropriate change in starting times. For this reason, the S units collect the information from the SC units (see Fig. 4). The corresponding S units will receive this redirecting information resulting in an inhibition ($+4 * -1$) of the S_{111} unit and an excitation ($+4 * 1$) of the S_{122} unit, thus working towards an acceptable solution. If these feedback weights are set correctly, feasible solutions will be generated without requiring explicit initialisation of the S units.

The resource constraints are implemented in the general structure presented in Fig. 3. The RC units collect information from the adequate S units and the Y units according to the resource equations. See Figures 3 and 4.

Suppose the starting time of operation 1 of job 1 on machine 1 (S_{111}) is 1, and the starting time of operation 2 of job 2 on machine 1 (S_{221}) is 2. In that case, unit Y_{121} receives a net input of $-1 + 2 = 1$ resulting in an activation of 1, signalling that S_{111} precedes operation S_{221} . The RC unit representing equation 5 receives -1 from S_{111} , 2 from unit S_{221} and -35 from Y_{121} . This value added to the bias (30) of this RC unit results in a net input of -5 resulting in an activation of 5, signalling a violation. The S_{111} and S_{221} units receive this activation through their weighted feedback connections, resulting in an advanced starting time of operation 111 and a delayed starting time of operation 122.

The RC unit representing equation 6 receives 1 from S_{111} , -2 from S_{221} , and 35 from Y_{121} . This value added to the bias (-3) of this RC unit results in a net input of 31, resulting in an activation of 0.

The network converged to an optimum solution with this 0 threshold of the S units, in 224 cycles. See Fig. 7.

Fig. 7. Optimum solution to 4/3/J/Cmax

8. CONCLUSIONS

It has been shown that an integer linear representation of a job-shop scheduling problem can be mapped on the proposed neural network structure.

Simulations have shown that optimum and near optimum solutions are found.

The way the network evolves to a solution does not require the 'rounded linear' programming method described by Foo and Takefuij (1988c) and van Hulle (1991). Due to this direct involvement of the binary variable representing units, sub-optimisation is largely avoided.

The selected fixed-order processing of the units results in a deterministic network. If the network has converged once, the performance of the structure can be determined. Random and simulated parallel processing result in non-deterministic networks. The noise introduced by these ways of processing might enhance the chance of finding an optimum solution.

Due to the carefully considered feedback connections, constraint violating initialisations result in feasible solutions. Obligatory optimal initialisation of the starting times as a result of sub-optimal feedback connections (Foo and Takefuij, 1988c; van Hulle, 1991) is not required. The proposed small values (0.1) of the weights of the feedback connections slows down the processing but increases the chance of finding an optimum or near-optimum solution.

Curtailing the search space by calculation and the setting of minimum starting times (thresholds) affects the calculation in both a positive and a negative way. Solutions are found more rapidly but the threshold may interfere with the evolution to an optimum solution.

Job-shop scheduling neural networks presented in the literature (Foo and Takefuij, 1988c; van Hulle, 1991), are modelled as an energy function. The simulation is performed by minimisation of this function according to an optimisation criterion. Only by modelling these representations as real neural networks with for instance Smallneurons, is their lack of a neural optimisation criterion revealed.

Solutions are found by a gradual parallel increase of starting times, resulting in a solution that satisfies the Cmax criterion. A powerful global optimisation criterion is not required for this application.

Other performance criteria might require the implementation of such an optimisation criterion. More research is required on how to implement these optimisation criteria in the neural structure.

The careful selection of the activation function of the S units renders the use of non-negativity constraints as proposed by van Hulle (1991) superfluous, resulting in a smaller neural network.

The design of the network doesn't guarantee an optimal solution for all problems. The quality of the solution can possibly be enhanced by the application of stochastic units, or by the application of heuristics (rules of thumb) in the determination of the feedback connections. The heuristics can be seen as a local optimisation criterion.

For future research it appears interesting to find out whether the performance of the proposed structure can be enhanced by the determination and implementation of a global optimisation criterion or heuristics as local optimisation criteria.

It has been shown that this type of network allows the rapid calculation of a solution to this np-complete problem, setting the stage for hardware developments, which may ultimately result in real-time job-shop schedulers for realistic problems.

9. ACKNOWLEDGEMENT

The authors are very grateful to Mr. P.J.F. Dupuis for the experiments and research performed for his master's thesis.

10. REFERENCES

- Baker, K.R. (1974). *Introduction to Sequencing and Scheduling*, Wiley, New York.
- Conway, R.W., Maxwell, W.L. and Miller, L.W. (1967). *Theory of Scheduling*, Reading, Addison-Wesley Publishing Company.
- Foo, Y.P.S. and Takefuij, Y. (1988a). Neural Networks for Solving Job-Shop Scheduling: Part 1. Problem Representation, *Proc. IEEE ICNN 88, San Diego, CA, July, II* 275-283.
- Foo, Y.P.S. and Takefuij, Y. (1988b). Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 2. Architecture and Simulations, *Proc. IEEE ICNN 88, San Diego, CA, July, II* 283-290.
- Foo, Y.P.S. and Takefuij, Y. (1988c). Integer Linear Programming Neural Networks for Job-Shop Scheduling, *Proc. IEEE ICNN 88, San Diego, CA, July, II* 341-348.
- French, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood, Chichester.
- Hopfield, J.J. and Tank, D.W. (1985). "Neural" Computation of Decisions in Optimization Problems, *Biol. Cybern.*, 52, 141-152.

- Hulle van, M.M. (1991). A Goal Programming Network for Mixed Integer Linear Programming: A case study for the job-shop scheduling problem, *Int. Journ. of Neural Systems*, Vol. 2, 201-209.
- Poliac, M.O., Lee, E.B., Slagle, J.R. and Wick, M.R. (1987). A Crew Scheduling Problem, *Proc. IEEE First Int. Conf. Neural Networks*, pp. 779-786.
- Rumelhart, D.E., and McClelland, J.L. (1986). *Parallel Distributed Processing: Explorations in the micro structure of cognition*, Vol 1: Foundations, MIT Press.
- Smit G.H. (1992). *A hierarchical control architecture for job-shop manufacturing systems*, Dissertation, Eindhoven University of Technology, Eindhoven.
- Tank, D.W., and Hopfield, J.J. (1986). Simple "Neural" optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit, *IEEE Trans. Circuits and Systems*, 33, 533-541.
- Willems, T.M., and Rooda, J.E. (1992). *Neural Nets, An introduction for System Engineers*, Memorandum WPA nr. 1343, Eindhoven University of Technology.
- Zhou, D.N., Cherkassky, V., Baldwin, T.R. and Hong, D.W. (1990). Scaling Neural Network for Job-Shop Scheduling, *Proc. IEEE International Joint Conference on Neural networks*, vol. 3.