TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Neural nets for job-shop scheduling, will they do the job?

Citation for published version (APA):
Rooda, J. E., & Willems, T. M. (1993). Neural nets for job-shop scheduling, will they do the job? In G. C. Goodwin, & R. J. Evans (Eds.), *12th IFAC triennial world congress on automatic control, Sydney, Australia, 18–23 July 1993, vol. 4* (pp. 349-352). Pergamon.

Document status and date:
Published: 01/01/1993

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.

Download date: 16. Nov. 2023

Copyright © IFAC 12th Triennial World Congress,
Sydney, Australia, 1993

NEURAL NETS FOR JOB-SHOP SCHEDULING, WILL THEY DO THE JOB?

T.M. Willems and J.E. Rooda

Faculty of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. A neural net 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 net structure. Use of the presented structure eliminated the need for integer adjustments. The search space was reduced by the use of precalculation, allowing the rapid calculation of feasible solutions. The neural net structure was reliable in simulated operation and its performance was superior to structures which have been presented previously.

Key Words. Job-shop scheduling, neural nets, 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. The problem with currently available techniques for solving this problem, is that they in general suffer from combinatorial explosion, making them less suitable for larger instances of np-complete problems.

Neural nets are 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 nets can be trained to exhibit specific behaviour (Rumelhart *et al.*, 1986), and they can also be predefined to perform constraint satisfaction / optimisation tasks (Tank and Hopfield, 1986). It has been shown (Foo and Takefuij, 1988a- c; Van Hulle, 1991) that constraint satisfaction and optimisation neural nets have the potential to solve the predictive job-shop scheduling problem in a satisfactorily rapid manner, but the neural nets presented in literature all suffer, to a greater or lesser degree, from design deficiencies. In this paper, a neural net structure is presented onto which an integer linear representation of a job-shop scheduling problem can be mapped. Simulation has shown that the net finds optimum and near-optimum solutions. Convergence is nearly deterministic, the only randomness being introduced by the random processing of the units. 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.

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

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

2. 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 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 predescribed, 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. In the literature, minimisation of the makespan is often used as the criterion, and it will also be used as criterion in this article too. 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 more abstract version of the job-shop scheduling problem is mostly encountered in the literature (e.g. Foo and Takefuij, 1988a-c; Van Hulle, 1991; Zhou *et al.*, 1990), and will be tackled here too. 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.

3. INTEGER LINEAR PROGRAMMING REPRESENTATION

The constraints can be translated to integer linear programming format (Baker, 1974). 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. The general sequence constraint has to be represented first. Suppose that the recipe of a job *i* states that operation *j* of job *i* requiring machine *b* is to be performed after operation *(j-1)* of job *i* requiring machine *a* is finished. Then, for a feasible schedule, it is necessary that

$$S_{ijb} - S_{i(j-1)a} - t_{i(j-1)a} >= 0 \qquad (1)$$

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*. If, on the other hand, job *p* precedes job *i* on machine *k* then operation *pjk* must be finished before job *i* can be processed on machine *k*. 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, stating whether job *i* precedes job *p* on machine *k* (value 1) or not (value 0). A constant *H* is added to ensure that one of the statements holds while the other is eleminated due to *H*.

$$S_{pjk} - S_{ijk} + H(1 - Y_{ipk}) - t_{ijk} >= 0 \qquad (2)$$

$$S_{ijk} - S_{pjk} + H * Y_{ipk} - t_{pjk} >= 0 \qquad (3)$$

with: $Y_{ipk} = 1$ *if* $S_{ijk} <= S_{pjk}$

with: $Y_{ipk} = 0$ *if* $S_{ijk} > S_{pjk}$

There are *nm(n-1)* resource constraint equations for a job-shop scheduling problem.

4. DESIGNING A NEURAL NET FOR THE JOB-SHOP SCHEDULING PROBLEM

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

The job-shop scheduling neural net 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 of these units is calculated by adding the previous activation (thus simulating a capacitor) to the summed incoming weighted activation. The activation function is of a deterministic linear threshold type. Since an operation can never start before time 0, the starting time of the entire schedule, the threshold can be set at 0, resulting in the implementation of a non-negativity constraint. The threshold 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 $0 + t_{ijk}$. 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 net.

SC- and RC units. The net input of these units is calculated by adding the bias to the summed incoming weighted activation. The activation function is of a deterministic negated linear threshold type.

Y units. The net input of the Y units is calculated by summation of the incoming weighted activation. The activation function is of a deterministic step type. The activation of this unit represents whether job *i* precedes job *j* on machine *k* or job *j* precedes job *i*.

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. First the connections to the SC units will be described. The constraint: $S_{ijb} - S_{i(j-1)a} - t_{i(j-1)a} >= 0$ is represented by a 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. 1. 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, requiring 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. 1.

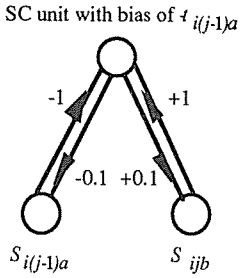

Fig. 1. General structure for sequence constraints

For the implementation of the resource constraints, 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. 2. The RC units should receive information from the Y units as prescribed by the equations. Furthermore the RC units have to receive information from the S units. The RCI unit representing equation 2 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 RCII unit representing equation 3 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. 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. Small weights of these connections diminish the chance of getting stuck in a local minimum, at the cost of longer processing times.

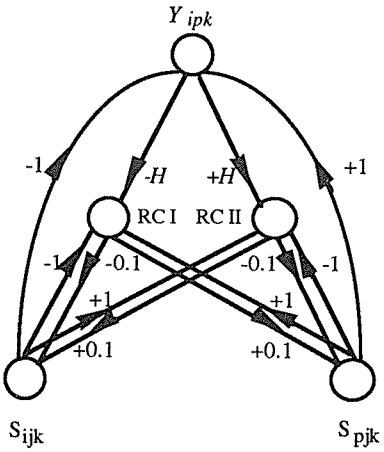

Fig. 2. General structure for resource constraints

5. RESULTS

The designed neural nets are modelled in the object oriented neural net modelling language 'Smallneurons' (Willems and Rooda, 1992). A 2/3/J/Cmax problem is solved by the proposed net. The (optimum) solution is obtained after 42 cycles, see Fig. 3.

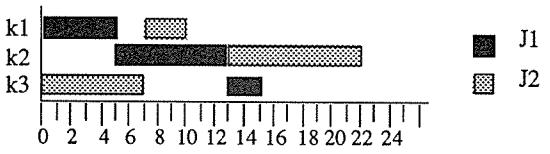

Fig. 3. Optimum solution to 2/3/J/Cmax

With the thresholds of the S units set at 0 (initial S_{ijk} values of 0), it took the net an average of 158 cycles to converge to the same solution.
In order to test the performance on a larger problem, a 4/3/J/Cmax problem was defined, see Tables 1 and 2.

Table 1.
Machine assignments

		Operation		
		1	2	3
Job	**1**	1	2	3
	2	2	1	3
	3	3	2	1
	4	2	3	1

Table 2.
Processing times

		Operation		
		1	2	3
Job	**1**	4	3	2
	2	1	4	4
	3	3	2	3
	4	3	3	1

The designed net finds a sub-optimal solution (see Fig. 4) after 142 cycles.

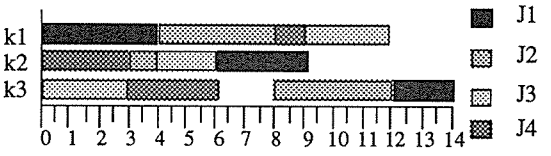

Fig. 4. Sub-optimal solution to 4/3/J/Cmax

The pre-set thresholds of the S units can result in the finding of a very good sub-optimal solution but may prevent the finding of the optimum solution. The optimum solution is presented in Fig. 5. Analysis of the solutions presented in Figures 4 and 5 shows that the quality of the solution is affected by the threshold values. The threshold value of S_{322} (3) is lower than the threshold value (4) of S_{122}, causing the initial setting of S_{322} preceding S_{122}. The feedback information from the SC- and RC units will only shift S_{122} and S_{322} relative to each other but will not exchange their positions.

Without precalculation, the thresholds of all S units are 0, only implementing a non-negativity constraint. The net converged to an optimum solution with this 0 threshold of the S units, in an average of 224 cycles. See Fig. 5.

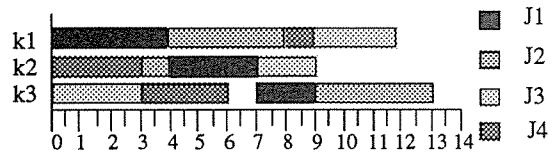

Fig. 5. Optimum solution to 4/3/J/Cmax

6. DISCUSSION

It has been shown that an integer linear representation of a job-shop scheduling problem can be mapped on the proposed neural net structure. Simulations have shown that optimum and near optimum solutions are found.
The way the net 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 (Y units), sub-optimisation is largely avoided.

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.

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.

Solutions are found by a gradual parallel increase of starting times, resulting in a solution that satisfies the Cmax criterion. A powerful general 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 net.

It has been shown that this type of net 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.

7. REFERENCES

Baker, K.R. (1974). *Introduction to Sequencing and Scheduling*, Wiley, New York.

Foo, Y.P.S. & Takefuji, Y. (1988). Stochastic 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. & Takefuji, Y. (1988). 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. & Takefuji, Y. (1988). 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

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

Rumelhart , D.E., & McClelland, J.L. (1986) *Parallel Distributed Processing: Explorations in the micro structure of cognition*, Vol 1: Foundations, MIT Press

Tank, D.W., & 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., & Rooda, J.E. (1992). *Neural Nets, An introduction for System Engineers*, Memorandum WPA nr. 1343, Eindhoven University of Technology