CONSTRAINT SATISFACTION ADAPTIVE NEURAL NETWORK AND EFFICIENT HEURISTICS FOR JOB-SHOP SCHEDULING

Shengxiang Yang [*,1] **Dingwei Wang** [*]

[*] *P. O. Box 135, Department of Systems Engineering, School of Information Science & Engineering, Northeastern University, Shenyang 110006, P. R. China*

Abstract: An efficient constraint satisfaction based adaptive neural network and heuristics hybrid approach for job-shop scheduling is presented. The adaptive neural network has the property of adaptively adjusting its connection weights and biases of neural units according to the sequence and resource constraints of job-shop scheduling problem while solving feasible solution. Two heuristics are used in the hybrid approach: one is used to accelerate the solving process of neural network and guarantee its convergence, the other is used to obtain non-delay schedule from solved feasible solution by neural network. Computer simulations have shown that the proposed hybrid approach is of high speed and excellent efficiency.

Keywords: Job-shop Scheduling, Constraint Satisfaction, Neural Networks, Heuristics

1. INTRODUCTION

Generally job-shop scheduling problems can be stated as follows (Conway *et al.*, 1967): given n jobs that have to be processed on m machines in a prescribed order under certain restrictive assumptions, the objective of job-shop scheduling is to decide how to arrange the processing orders and starting times of operations sharing the same machine for each machine, in order to optimize certain criteria, e.g., minimize the makespan. Generally, job-shop scheduling belongs to the large class of NP-complete problems. It is very hard to find its optimal solution. An NP-complete problem exhibits an exponential growth in the computation time as the size of the problem increases linearly. Researchers turned to search its near-optimal solutions to meet practical need with all kind of heuristic algorithms (French, 1982).

Ever since Foo and Takefuji (Foo and Takefuji, 1988*a*; Foo and Takefuji, 1988*b*) first used neural network to solve job-shop scheduling problem, several neural network architectures have been presented to solve job-shop scheduling problem, for details see (Zhou *et al.*, 1989; Willems and Brandts, 1995). All above mentioned neural networks are basely non-adaptive networks with the connection weights and biases prescribed in advance before the networks begin to work.

An efficient constraint satisfaction adaptive neural network (CSANN) and heuristics hybrid approach for job-shop scheduling is proposed in this paper. CSANN has the property of easily mapping the constraints of scheduling problem into its architecture. Meanwhile CSANN has the property of adaptively adjusting its weights of connections and biases of neural units according to the actual situation of constraint violations during its

[1] Supported by the National Nature Science Foundation (No. 69684005) and National High-Tech Program (No. 863-511-9609-003) of P. R. China.

processing to remove these violations for obtaining feasible solutions. Thus CSANN is constraint satisfaction based and adaptive. The adaptive property of CSANN makes it different from other neural network for job-shop scheduling problems and results in a simpler network architecture. In the proposed hybrid approach, CSANN is used to obtain feasible solutions, one of two heuristics is used to accelerate the solving process of CSANN and guarantee obtaining feasible solutions, the other is used to obtain non-delay solution from feasible solution solved by CSANN. The computer simulations have shown that the proposed hybrid approach has good performance as to the quality of solutions and the solving speed.

2. FORMULATION OF JOB-SHOP SCHEDULING PROBLEM

Generally for job-shop scheduling problem there are two types of constraints: *sequence constraint* and *resource constraint*. The first type states that two operations of a job cannot be processed at the same time. The second states that no more than one job can be performed on one machine at the same time. Job-shop scheduling can be viewed as an optimization problem, bounded by both sequence and resource constraints.

Denote $N = \{1, \cdots, n\}$ and $M = \{1, \cdots, m\}$, where n and m are the numbers of jobs and machines. Let n_i be the operation number of job i. O_{ikq} represents operation k of job i on machine q, S_{ikq} and T_{ikq} represent the starting time and processing time (which is known in advance) of O_{ikq}, S_{ie_iq} and T_{ie_iq} represent the starting time and processing time of the last operation of job i respectively. Denote r_i and d_i as the release date (earliest starting time) and due date (latest ending time) of job i. Let P_i denote the set of operation pairs $[O_{ikp}, O_{ilq}]$ with precedence restriction of job i, where operation O_{ikp} must precede operation O_{ilq}. Let R_q be the set of operations O_{ikq} that will be processed on machine q.

Taking minimizing the makespan as the criterion, the mathematical programming formulation of the considered job-shop scheduling problem is presented as follows:

$$Minimize \;\; E = max_{i \in N}(S_{ie_iq} + T_{ie_iq})$$

subject to

$$\begin{gathered} S_{ilq} - S_{ikp} \geq T_{ikp}, \\ [O_{ikp}, O_{ilq}] \in P_i, \;\; k, l \in \{1, \cdots, n_i\} \end{gathered} \tag{1}$$

$$\begin{gathered} S_{jlq} - S_{ikq} \geq T_{ikq} \; or \; S_{ikq} - S_{jlq} \geq T_{jlq}, \\ O_{ikq}, O_{jlq} \in R_q, \;\; i, j \in N, \;\; q \in M \end{gathered} \tag{2}$$

$$\begin{gathered} r_i \leq S_{ijq} \leq d_i - T_{ijq}, \\ i \in N, \;\; j \in \{1, \cdots, n_i\}, \;\; q \in M \end{gathered} \tag{3}$$

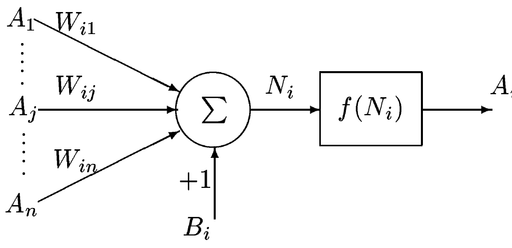

Fig. 1. General neural unit model

where the cost function is the ending time of the latest operation, *i.e.* maximal complete time of job-shop scheduling problem. Equation (1) represents the sequence constraint; Equation (2), in a disjunctive type, represents resource constraints; Equation (3) represents the release date and due date constraints.

3. MODEL OF CONSTRAINT SATISFACTION ADAPTIVE NEURAL NETWORK

3.1 *Neural units of CSANN*

Generally neural unit consists of two parts: a linear summator and a nonlinear activation function which are serialized (see Fig. 1). The summator of unit i receives all activations $A_j (j = 1, \cdots, n)$ from connected units and sums the received activations, weighted with connection weight W_{ij}, together with a bias B_i. The output of summator is the net input N_i, this net input N_i is passed through an activation function $f(.)$, resulting in the activation A_i of unit i. The summator and the activation function are defined as follows:

$$A_i = f(N_i) = f(\sum_{j=1}^{n}(W_{ij} \times A_j) + B_i) \tag{4}$$

where W_{ij} is the connection weight from unit j to unit i.

Based on the general neural unit, CSANN contains three kinds of units: *ST-units*, *SC-units* and *RC-units*. The first kind of units represent the starting times of all operations. Each ST-unit represents one operation of job-shop scheduling problem with its activation representing the starting time of the operation. The second represent whether the sequence constraints are violated. The third represent whether the resource constraints are violated.

The net input of a ST-unit ST_i is calculated by

$$\begin{aligned} N_{ST_i}(t) = & \sum_j (W_{ij} \times A_{SC_j}(t)) + \\ & \sum_k (W_{ik} \times A_{RC_k}(t)) + A_{ST_i}(t-1) \end{aligned} \tag{5}$$

where the net input of unit ST_i is summed from three parts. The first part comes from the weighted activations of SC-units connected with ST_i, which implements feedback adjustments because of sequence violations. The second part comes from the weighted activations of RC-units connected with ST_i, implementing feedback adjustments because of resource violations. The third part comes from the previous activation, with weight being $+1$, of unit ST_i itself.

The activation function of ST-units is a deterministic linear-segmented function as follows.

$$A_{ST_i}(t) = \begin{cases} r_i, \ N_{ST_i}(t) < r_i \\ N_{ST_i}(t), \\ \quad r_i \leq N_{ST_i}(t) \leq d_i - T_{ST_i} \\ d_i - T_{ST_i}, \\ \quad N_{ST_i}(t) > d_i - T_{ST_i} \end{cases} \quad (6)$$

where r_i and d_i are the release date and due date of job i to which the operation, corresponding to unit ST_i, belongs. T_{ST_i} is the processing time of the operation corresponding to unit ST_i. This activation function implements the release date and due date constraints described by equation (3).

The SC-units receive the incoming weighted activations from the connected ST-units, representing operations of the same job. The RC-units receive the incoming weighted activations from the connected ST-units, representing operations to be processed on the same machine. The net input of a SC-unit or RC-unit has the same definition form as follows:

$$N_{C_i}(t) = \sum_j (W_{ij} \times A_{ST_j}(t)) + B_{C_i} \quad (7)$$

where C_i equals SC_i or RC_i, and B_{C_i} is the bias of neural unit SC_i or unit RC_i. The bias B_{C_i} is added to the incoming weighted activations of the connected ST-units ST_j's and equals the processing time of a relative operation, formulated in this equation.

The activation function of a SC-unit or a RC-unit is a deterministic linear-segment function, defined as follows.

$$A_{C_i}(t) = \begin{cases} 0, & N_{C_i}(t) \geq 0 \\ -N_{C_i}(t), & N_{C_i}(t) < 0 \end{cases} \quad (8)$$

The activation of a SC-unit or RC-unit being greater than zero means the corresponding sequence constraint or resource constraint is violated and there are feedback adjustments from this SC-unit or RC-unit to connected ST-units through adaptive weighted connections.

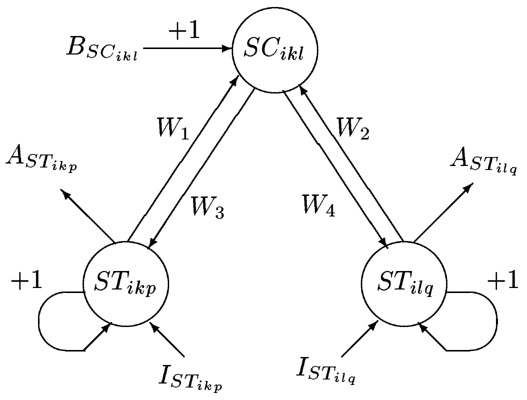

Fig. 2. SC-block unit

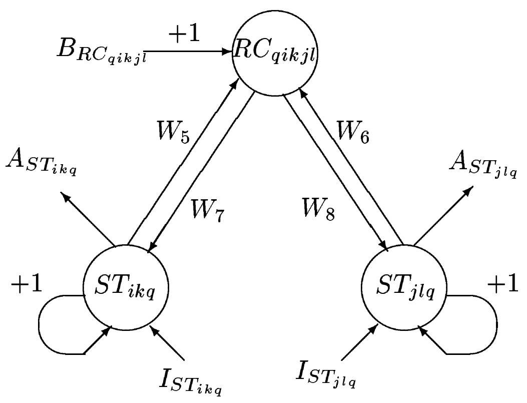

Fig. 3. RC-block Unit

3.2 *Adaptive weights and biases*

All units of CSANN are connected according to the two kinds of sequence and resource constraints of specific job-shop scheduling problem, resulting in two blocks: SC-block (sequence constraints block) and RC-block (resource constraints block). Each unit of SC-block contains two ST-units, responding to two operations of a job, and one SC-unit, representing whether the sequence constraint between these two operations is satisfied (see Fig.2). Each unit of RC-block contains two ST-units, responding to two operations sharing the same machine, and one RC-unit, representing whether the resource constraint between these two operations is satisfied (see Fig.3).

Fig. 2 presents an example of SC-block unit, denoted by SCB_{ikl}. ST-units ST_{ikp} and ST_{ilq} represent two operations O_{ikp} and O_{ilq} of job i. Their activations $A_{ST_{ikp}}$ and $A_{ST_{ilq}}$ represent the starting times S_{ikp} and S_{ilq} of O_{ikp} and O_{ilq}. The SC-unit SC_{ikl} represents whether the sequence constraint of equation (1) between O_{ikp} and O_{ilq} is violated, with $B_{SC_{ikl}}$ being its bias. The weights and bias are valued as follows:

$$\begin{aligned} &W_1 = -1, \ W_2 = 1, \ W_3 = -W, \ W_4 = W, \\ &B_{SC_{ikl}} = -T_{ikp} \end{aligned} \quad (9)$$

where W is a positive feedback adjustment parameter (the same with following equations where W appears).

If violation exists at time t, $N_{SC_{ikl}}(t) = A_{ST_{ilq}}(t) - A_{ST_{ikp}}(t) - T_{ikp} < 0$, the activation of SC_{ikl} is calculated by

$$\begin{aligned} A_{SC_{ikl}}(t) &= A_{ST_{ikp}}(t) + T_{ikp} - A_{ST_{ilq}}(t) \\ &= S_{ikp}(t) + T_{ikp} - S_{ilq}(t) \end{aligned} \quad (10)$$

and the feedback adjustments from SC_{ikl} to ST_{ikp} and ST_{ilq} are shown as follows:

$$S_{ikp}(t+1) = S_{ikp}(t) - W \times A_{SC_{ikl}}(t) \quad (11)$$

$$S_{ilq}(t+1) = S_{ilq}(t) + W \times A_{SC_{ikl}}(t) \quad (12)$$

From the above equations we can see the feedback adjustments from unit SC_{ikl} puts back the starting time S_{ikp} of operation O_{ikp} in time axis, while putting forward S_{ilq} of O_{ilq}. Thus the sequence violation between O_{ikp} and O_{ilq} can be removed.

Fig. 3 presents an example of RC-block unit, denoted by RCB_{qikjl}, representing the resource constraint between O_{ikq} and O_{jlq} on machine q. At time t during the processing of network, the weights and bias are adaptively valued as following two cases show.

Case 1: If $S_{ikq}(t) \leq S_{jlq}(t)$, equation (13) holds

$$\begin{aligned} &W_5 = -1,\ W_6 = 1,\ W_7 = -W,\ W_8 = W, \\ &B_{RC_{qikjl}} = -T_{ikq} \end{aligned} \quad (13)$$

In this case RCB_{qikjl} represents a sequence constraint described by the first disjunctive equation of equation (2). If violation exists, the activation of RC_{qikjl} and the feedback adjustments from RC_{qikjl} to ST_{ikq} and ST_{jlq} are calculated by

$$\begin{aligned} A_{RC_{qikjl}}(t) &= A_{ST_{ikq}}(t) + T_{ikq} - A_{ST_{jlq}}(t) \\ &= S_{ikq}(t) + T_{ikq} - S_{jlq}(t) \end{aligned} \quad (14)$$

$$S_{ikq}(t+1) = S_{ikq}(t) - W \times A_{RC_{qikjl}}(t) \quad (15)$$

$$S_{jlq}(t+1) = S_{jlq}(t) + W \times A_{RC_{qikjl}}(t) \quad (16)$$

Case 2: If $S_{ikq}(t) \geq S_{jlq}(t)$, equation (17) holds

$$\begin{aligned} &W_5 = 1,\ W_6 = +1,\ W_7 = W,\ W_8 = -W, \\ &B_{RC_{qikjl}} = -T_{jlq} \end{aligned} \quad (17)$$

In this case RCB_{qikjl} represents a sequence constraint described by the second disjunctive equation of equation (2). If there exists violation, the activation of RC_{qikjl} and the feedback adjustments are calculated by

$$\begin{aligned} A_{RC_{qikjl}}(t) &= A_{ST_{jlq}}(t) + T_{jlq} - A_{ST_{ikq}}(t) \\ &= S_{jlq}(t) + T_{jlq} - S_{ikq}(t) \end{aligned} \quad (18)$$

$$S_{ikq}(t+1) = S_{ikq}(t) + W \times A_{RC_{qikjl}}(t) \quad (19)$$

$$S_{jlq}(t+1) = S_{jlq}(t) - W \times A_{RC_{qikjl}}(t) \quad (20)$$

4. HEURISTICS AND HYBRID APPROACH

4.1 *Heuristics*

Two heuristics are used as improvement tool of CSANN for job-shop scheduling problems. One is used to accelerate the solving process of CSANN and guarantee feasible solutions, the other is used to obtain non-delay schedule from feasible solution obtained by CSANN.

Heuristics 1: Exchange the orders of two near operations. This heuristics has two folds of function: to accelerate the solving process and to guarantee feasible solution. The former is for two near operations coming from the same job, while the latter is for two near operations sharing the same machine.

On the one hand, assuming $[O_{ikp}, O_{ilq}] \in P_i$. In order to accelerate the solving speed of CSANN, at time t during its processing, if $S_{ikp}(t) \geq S_{ilq}(t)$, exchange the orders of O_{ikp} and O_{ilq} by exchanging their starting times as follows:

$$S_{ikp}(t+1) = S_{ilq}(t),\ S_{ilq}(t+1) = S_{ikp}(t) \quad (21)$$

In fact equation (21) is a more direct method of removing sequence violation than that of the feedback adjustment of CSANN. Thus the adjustment time from removing sequence violations may be shortened and the solving process of CSANN for feasible solution is accelerated.

On the other hand, during the processing of CSANN there may appear the phenomenon of "dead lock" which can result in no feasible solution. In order to remove "dead lock", we use the following heuristic: exchange the orders of two near operations sharing the same machine by exchanging their starting times.

Assuming O_{ikq} and $O_{ijq} \in R_q$, during the processing of CSANN, if $T_{qikjl}(t) \geq T$, the following equation begins to work:

$$S_{ikq}(t+1) = S_{jlq}(t),\ S_{jlq}(t+1) = S_{ikq}(t) \quad (22)$$

where variable $T_{qikjl}(t)$ is the summed times that operation pairs O_{ikq} and O_{jlq} have their starting times continuously changed, keeping the same adjusting effects, at time t ever since the previous zero-reset because of resource conflict on machine q during the processing of CSANN. The parameter T is a prescribed positive integer.

The above heuristic can be used together with CSANN to guarantee the feasible solution. The phenomenon of "Dead lock" results from the conflicts of feedback adjustments while removing sequence and resource constraint violations. For example, assuming $[O_{ikp},\ O_{ilq}] \in P_i$ and O_{ilq}, $O_{jmq} \in R_q$. During the processing of CSANN, the

SC-unit SC_{ikl} may put forward the starting time S_{ilq} of operation O_{ilq} along the positive direction of time axis through feedback adjustment because of sequence violation, while the RC-unit RC_{qiljm} may put back S_{ilq} through feedback adjustment because of resource violation. Thus there may exist conflicts resulting from this two kinds of adjustments which result in "dead lock". "Dead lock" results in the nonconvergence of CSANN to its stable station, which corresponds to the feasible solution of specific job-shop scheduling problem. By using proposed heuristic, when the phenomenon of "dead lock" happens and S_{ilq} has been continuously put back T times because of resource violation between O_{ilq} and O_{jmq}, that is, at time t $T_{qikjl}(t)$ reaches T, the starting time S_{ilq} of O_{ilq} may be exchanged with S_{jmq} of O_{jmq}. Thus "dead lock" can be effectively avoided and feasible solution is guaranteed.

Heuristics 2: Obtain non-delay schedule from feasible solution solved by CSANN. A schedule is *non-delay* if no machine lies idle when there is at least one job waiting to be operated on that machine (French, 1982). A non-delay schedule is a local optimal schedule with orders of operations to be operated on each machine already determined. A schedule is *active* if no operation can be started earlier without delaying another operation or violating the sequence constraints. It is evident that an optimal schedule is an active one. The set of non-delay schedules is a proper subset of the active ones. CSANN can obtain feasible solution quickly, but there may be many idle times for each machine with operations available to be operated. Obviously these idle times heavily degrade the quality of feasible schedule and should be compacted away in order to shorten makespan or improve the quality of schedule. The detailed heuristics is as follow.

Assuming a feasible solution $\{S_{ikp}, i \in N, k \in \{1, \ldots, n_i\}, p \in M\}$ has been obtained by CSANN. Sort them in non-decreasing order. Then from the minimal to the maximal, each S_{ikp} is adjusted as follows:

$$S'_{ikp} = \begin{cases} S_{ljp} + T_{ljp}, \\ \quad S_{ljp} + T_{ljp} \geq S_{i(k-1)q} + T_{i(k-1)q} \\ S_{i(k-1)q} + T_{i(k-1)q}, \\ \quad S_{ljp} + T_{ljp} < S_{i(k-1)q} + T_{i(k-1)q} \end{cases} \tag{23}$$

where S'_{ikp} is the starting time of O_{ikp} in the obtained non-delay schedule after the heuristics is run. $O_{i(k-1)q}$ is the precedence operation of O_{ikp} from the same job i, and O_{ljp} is the precedence operation of O_{ikp} sharing the same machine p. Equation (23) means to shorten each starting time S_{ikp} to the completion time of $O_{i(k-1)q}$ or the completion time of O_{ljp}. The adjustments of all starting times are dynamic, *i.e.*, the starting time of previous operation that has been adjusted works while adjusting the latter operations. For example, supposing that S_{ikp} has been adjusted into S'_{ikp}, when computing $S'_{i(k+1)q}$ of operation $O_{i(k+1)q}$ which is just next to O_{ikp} of the same job i, S'_{ikp} is used in equation (23) instead of S_{ikp}. Thus each operation needs once and only once adjustment to obtain non-delay schedule. With heuristics 2 used, the optimal schedule may be achieved when obtained non-delay schedule fall into the optimal set.

4.2 *Main steps of hybrid approach*

The main steps of hybrid Approach are as follows:

Step 1: Build up CSANN model, set parameter values for T and W, prescribe the expected makespan;

Step 2: Randomly or by hand initialize the starting time $S_{ikp}(0)$ for each operation O_{ikp} as the initial net input $I_{ST_{ikp}}$ of each ST-unit ST_{ikp};

Step 3: Run each SC-unit SC_{ikl} of SC-block, calculate its activation with equation (10). $A_{SC_{ikl}}(t) \neq 0$ means the dissatisfaction of sequence constraint, then adjust activations of relative ST-units with equations (11, 12) or with equations (21) under the condition of heuristic 1;

Step 4: Run each RC-unit RC_{qikjl} of RC-block, calculate its activation with equation (14) or (18). $A_{RC_{qikjl}}(t) \neq 0$ means the dissatisfaction of resource constraint corresponding to equation (2). Then adjust $S_{ikq}(t+1)$ and $S_{jlq}(t+1)$ with equations (15, 16) or equations (19, 20), or with equations (22) under the condition of heuristic 1;

Step 5: Repeat step 3 and step 4 until all units are in stable states without changes, which means that all the sequence and resource constraints are satisfied and the feasible solution is obtained.

Step 6: If heuristics 2 is not used, stop now; Otherwise, use equation (23) to obtain non-delay schedule. When the non-delay schedule is obtained, stop the program.

In the practical process of hybrid strategy, expected makespan is usually used as the due dates of all jobs, which maybe a number greater than the minimal makespan of solved problem.

5. SIMULATION STUDY

Example: Table 5 presents a $6/6/J/C_{max}$ problem, where (M, T) represents the relevant operation will be processed on machine M with processing time being T. And the sequence constraints of

Table 1 An example of $6/6/J/C_{max}$ problem

Operation No.	1	2	3	4	5	6
Job 1	3,1	1,3	2,6	4,7	6,3	5,6
Job 2	2,8	3,5	5,10	6,10	1,10	4,4
Job 3	3,5	4,4	6,8	1,9	2,1	5,7
Job 4	2,5	1,5	3,5	4,3	5,8	6,9
Job 5	3,9	2,3	5,5	6,4	1,3	4,1
Job 6	2,3	4,3	6,9	1,10	5,4	3,1

Table 2 Simulation results by hybrid approach

Expected makespan	Initial starting times	Makespan (E) Ave/Min/Max	Runtime (Seconds) Ave/Min/Max
200	0	76	1
200	Random	76/60/94	0.17/0/1
100	0	69	1
100	Random	75/60/93	0.21/0/1
58	0	56	30
58	Random	57/55/58	6.6/1/44

all jobs are the same: from operation 1 to operation 6 orderly. This example has the optimum, *i.e.* minimal makespan, of 55.

The simulations are completed on a Pentium 586/133 PC and under VC++ development environment. In the simulations before the hybrid approach begins to run scheduler can prescribe an expected makespan, which can be used as the common due date for all jobs. The simulations are finished with expected makespan prescribed to be 200 (big enough for feasible solution, greater than the sum of processing times of all operations, 197), 100 (appropriate value) and 58 (near-optimal value) respectively. For each expected makespan, 100 experiments are carried out, of which the first one is executed under zero initial condition with initial starting times of all operations set to zero, the other 99 experiments are carried out with initial starting times of all operations valued in a randomly uniform distribution between [0,100]. For all experiments, the parameters are valued as follows: $T = 5$, $W = 0.5$ and $X = 5$, and the release dates for all jobs are set to zero.

Table 2 shows the simulation results with respect to average, minimum and maximum of obtained makespans and program run times for each prescribed expected makespan respectively. In table 2 the run time being zero means that it is less than one zero. Fig.4 shows the Gantt chart of an obtained optimal solution, where (i, j) means the relative operation is the jth operation of job i.

6. CONCLUSIONS

Simulations show that the proposed hybrid approach for job-shop scheduling problems has very good performance with respect to the quality of solution and the speed of calculation. From table 2 and Fig.4 we can see: Given zero initial solution, hybrid approach can find good or near-

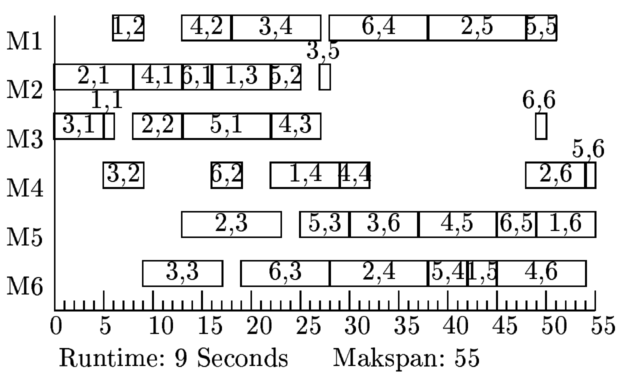

Fig. 4. An optimal solution

optimal schedules for different expected makespan restriction; Given appropriate expected makespan restriction, hybrid approach can always find good, near-optimal or optimal schedules; The solving speed of hybrid approach is very high.

While the proposed hybrid approach is used for solving practical job-shop scheduling problems, we can run the hybrid approach to solve the practical problem several times, *e.g.* 10 times, with appropriate expected makespan restriction. Then obtain the solution with shortest makespan. Usually the obtained best solution is a near-optimal or optimal solution of the problem. Thus we can use it as practical schedule.

7. REFERENCES

Conway, R. W., W.L. Maxwell and L.W. Miller (1967). *Theory of Scheduling.* Reading, MA: Addison-Wesley.

Foo, S. Y. and Y. Takefuji (1988*a*). Neural networks for solving job-shop scheduling: Part 1. problem representation. *Proc. IEEE IJCNN* **II**, 275–282.

Foo, S. Y. and Y. Takefuji (1988*b*). Stochastic neural networks for solving job-shop scheduling: Part 2. architecture and simulations. *Proc. IEEE IJCNN* **II**, 283–290.

French, S. (1982). *Sequencing and scheduling: an introduction to the mathematics of the job-shop.* New York: Wiley.

Willems, T. M. and L. E. M. W. Brandts (1995). Implementing heuristics as an optimization criterion in neural networks for job-shop scheduling. *Journal of Intelligent Manufacturing* **6**, 377–387.

Zhou, D. N., V. Charkassky, T. R. Baldwin and D. W. Hong (1989). Scaling neural network for job-shop scheduling. *Proc. IEEE IJCNN* **3**, 889–894.