GENETIC ALGORITHM AND NEURAL NETWORK HYBRID APPROACH FOR JOB-SHOP SCHEDULING

KAI ZHAO, SHENGXIANG YANG AND DINGWEI WANG

Northeastern University, Shenyang, Liaoning, 110006, P. R. China, Email: xmliu@ramm.neu.edu.cn

Abstract This paper proposes a genetic algorithm (GA) and constraint satisfaction adaptive neural network (CSANN) hybrid approach for job-shop scheduling problems. In the hybrid approach, GA is used to iterate for searching optimal solutions, CSANN is used to obtain feasible solutions during the iteration of genetic algorithm. Simulations have shown the valid performance of the proposed hybrid approach for job-shop scheduling with respect to the quality of solutions and the speed of calculation.

Keywords job-shop scheduling, genetic algorithm, neural network

1. INTRODUCTION

Generally deterministic job-shop scheduling problems can be stated as follows [1]: 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. Job-shop scheduling belongs to the large class of NP-hard problem, it is very hard to find its optimal solution. Researchers turned to search its near-optimal solutions to meet practical need with all kind of heuristic algorithms [2]. More recently GAs have been used to solve job-shop scheduling problems [3, 4]. Ever since Foo and Takefuji [5] first used neural network to solve job-shop scheduling problem. After that, several neural network architectures have been presented for job-shop scheduling [6, 7]. All these neural networks are basely non-adaptive networks.

In this paper we propose a new hybrid approach of GA and CSANN to solve job-shop scheduling problem. In the hybrid approach, GA is used to iterate for searching optimal solutions, CSANN is used to obtain feasible solutions during the iteration of genetic algorithm. CSANN has the property of easily mapping the constraints of scheduling problem into its architecture and removing the violations of the mapped constraints during its processing. 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 to remove these violations. Simulations have shown that the hybrid approach has good performance with respect to the quality of solutions and the speed of calculation.

2. 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. Different manufacturing systems require different optimization criteria, such as stock size, mean lead time and makespan. Minimization of the makespan will be used in this paper.

Denote $N=\{1,\cdots,n\}$, $M=\{1,\cdots,m\}$. Let n_i be the operation number of job *i*. Let O_{ikq} represent operation *k* of job *i* on machine *q*, S_{ikq} and T_{ikq} represent the starting time and processing time 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. The processing time of each operation is known and fixed. Denote r_i and d_i as the release and due date of job *i*. P_i denotes the set of operation pairs $[O_{ikp}, O_{ilq}]$ where operation O_{ikp} must precede operation O_{ilq} of job *i*. Let R_q be the set of all operations on machine *q*. The mathematical formulation is presented as follows:

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

s.t.

$$S_{ilq}-S_{ikp}\geq T_{ikp},\quad [O_{ikp},O_{ilq}]\in P_i,\ k,l\in\{1,\cdots,n_i\},i\in N \qquad (1)$$

$$S_{jlq}-S_{ikq}\geq T_{ikq}\ or\ S_{ikq}-S_{jlq}\geq T_{jlq},\quad O_{ikq},O_{jlq}\in R_q,\ i,j\in N,q\in M \qquad (2)$$

$$r_i\leq S_{ijq}\leq d_i-T_{ijq},\quad i\in N,\ j\in\{1,\cdots,n_i\},q\in M \qquad (3)$$

where 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. The cost function is the ending time of the latest operation, *i.e.*, maximal complete time of job-shop scheduling problem.

3. MODEL OF CSANN

3.1 UNITS OF CSANN

Generally neural unit consists of two parts: a linear summator and a nonlinear activation function which are serialized. 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 is passed through an activation function $f(\cdot)$, resulting in the activation A_i of unit *i*. The summator and the activation function are defined as : $A_i = f(N_i) = f(\sum_{j=1}^{n}(W_{ij} * A_j) + B_i)$, 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: *S-units*, *SC-units* and *RC-units*. The first kind of units represent the starting times of all operations. Each S-unit represents one operation of job-shop scheduling problem with 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 S-unit, *e.g.* SU_i, is calculated by

$$N_{SU_i}(t) = \sum_j (W_{ij} * A_{SC_j}(t)) + \sum_k (W_{ik} * A_{RC_k}(t)) + A_{SU_i}(t-1) \quad (4)$$

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

The activation function of S-units is a deterministic linear-segmented function as follows:

$$A_{SU_i}(t) = \begin{cases} r_i & N_{SU_i}(t) < r_i \\ N_{SU_i}(t) & r_i \le N_{SU_i}(t) \le d_i - T_{SU_i} \\ d_i - T_{SU_i} & N_{SU_i}(t) > d_i - T_{SU_i} \end{cases} \quad (5)$$

where r_i and d_i are the release date and due date of job *i* to which the operation corresponding to SU_i belongs. T_{SU_i} is the processing time of the operation corresponding to unit SU_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 S-units, representing operations of the same job. The RC-units receive the incoming weighted activations from the connected S-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:

$$N_{C_i}(t) = \sum_j (W_{ij} * A_{SU_j}(t)) + B_{C_i} \quad (6)$$

where C_i equals SC_i or RC_i, B_{C_i} is the bias of unit SC_i or unit RC_i. The bias B_{C_i} is added to the incoming weighted activations of connected units and equals the processing time of relative operation.

The activation function of a SC-unit or RC-unit is a linear-segmented function as follows.

$$A_{C_i}(t) = \begin{cases} 0 & N_{C_i}(t) \ge 0 \\ -N_{C_i}(t) & N_{C_i}(t) < 0 \end{cases} \quad (7)$$

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 S-units through adaptive weighted connections.

3.2 ADAPTIVE CONNECTIONS

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 S-units, responding to two operations of a job, and one SC-unit, representing whether the sequence constraint between these two operations is violated (see Fig.1). Each unit of RC-block contains two S-units, responding to two operations sharing the same machine, and one RC-unit, representing whether the resource constraint between these two operations is violated (see Fig.2).

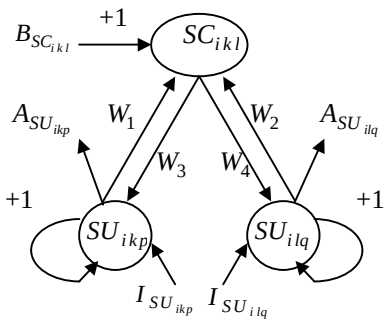

Fig.1 SC-BLOCK UNIT

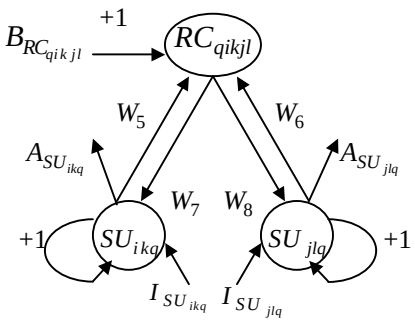

Fig.2 RC-BLOCK UNIT

Fig.1 presents an example of SC-block unit, denoted by SCB_{ikl} . S-units SU_{ikp} and SU_{ilq} represent two operations O_{ikp} and O_{ilq} of job *i*. Their activations $A_{SU_{ikp}}$ and $A_{SU_{ilq}}$ 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:

$$W_1 = -1, W_2 = 1, W_3 = -W, W_4 = W, B_{SC_{ikl}} = -T_{ikp} \quad (8)$$

where *W* is positive feedback adjustment parameter (the same where *W* appears latterly). If violation exists at time *t*, the activation of SC_{ikl} is calculated by

$$\begin{aligned} A_{SC_{ikl}}(t) &= A_{SU_{ikp}}(t) + T_{ikp} - A_{SU_{ilq}}(t) \\ &= S_{ikp}(t) + T_{ikp} - S_{ilq}(t) \end{aligned} \quad (9)$$

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

$$S_{ikp}(t+1) = S_{ikp}(t) - W * A_{SC_{ikl}}(t) \quad (10)$$

$$S_{ilq}(t+1) = S_{ilq}(t) + W * A_{SC_{ikl}}(t) \quad (11)$$

Above equations show that the feedback adjustments from SC_{ikl} puts back the starting time S_{ikp} of O_{ikp} in time axis, while putting forward S_{ilq} . Thus the sequence violation between O_{ikp} and O_{ilq} can be removed.

Fig.2 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 $S_{ikq}(t) \le S_{jlq}(t)$, equation (12) holds.

$$W_5 = -1, W_6 = 1, W_7 = -W, W_8 = W, B_{RC_{qikjl}} = -T_{ikq} \quad (12)$$

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 feedback adjustments are calculated by

$$\begin{aligned} A_{RC_{qikjl}}(t) &= A_{SU_{ikq}}(t) + T_{ikq} - A_{SU_{jlq}}(t) \\ &= S_{ikq}(t) + T_{ikq} - S_{jlq}(t) \end{aligned} \quad (13)$$

$$S_{ikq}(t+1) = S_{ikq}(t) - W * A_{RC_{qikjl}}(t) \quad (14)$$

$$S_{jlq}(t+1) = S_{jlq}(t) + W * A_{RC_{qikjl}}(t) \quad (15)$$

Case 2: If $S_{ikq}(t) \ge S_{jlq}(t)$, equation (16) holds.

$$W_5 = 1, W_6 = -1, W_7 = W, W_8 = -W, B_{RC_{qikjl}} = -T_{jlq} \quad (16)$$

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

$$A_{RC_{qikjl}}(t) = S_{jlq}(t) + T_{jlq} - S_{ikq}(t) \quad (17)$$

$$S_{ikq}(t+1) = S_{ikq}(t) + W * A_{RC_{qikjl}}(t) \quad (18)$$

$$S_{jlq}(t+1) = S_{jlq}(t) - W * A_{RC_{qikjl}}(t) \quad (19)$$

3.3 SOLVING STEPS OF CSANN

Step 1: Build up CSANN model, set H and W values;

Step 2: Initialize the starting time $S_{ikp}(0)$ for each operation O_{ikp} as the initial net input $I_{SU_{ikp}}$ of each S-unit SU_{ikp} ;

Step 3: Run each SC-unit SC_{ikl} of SC-block, calculate its activation with equation (9). $A_{SC_{ikl}}(t) \ne 0$ means the dissatisfaction of sequence constraint, then adjust activations of relative S-units with equations (10, 11);

Step 4: Run each RC-unit RC_{qikjl} of RC-block, calculate its activation with equation (13) or (17). $A_{RC_{qikjl}}(t) \ne 0$ means the dissatisfaction of resource constraint corresponding to equation (2). Then adjust $S_{ikp}(t+1)$ and $S_{ilq}(t+1)$ with equations (14, 15) or equations (18, 19) or with equations (20);

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

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} , $O_{jlq} \in R_q$, during the processing of CSANN, if $H_{qikjl}(t) \ge H$, the following two equations work.

$$S_{ikq}(t+1) = S_{jlq}(t), \quad S_{jlq}(t+1) = S_{ikq}(t) \quad (20)$$

where variable $H_{qikjl}(t)$ is the summed times that operation pairs O_{ikq} and O_{jlq} have their starting times changed continuously with the same adjusting effects because of resource conflict on machine *q* at time *t* ever since the previous zero-reset. *H* is a positive integer. With above heuristic, “dead lock” can be effectively avoided.

4. HYBRID APPROACH

4.1 GENETIC ALGORITHM

The main components of proposed GA are as follows:

Encoding mode: Each chromosome is formed of several subchromosomes, one for each machine. The length of chromosome is the total operations of all jobs. Each subchromosomes is formed of natural number string, each number identifying the job number to which the operation that has to be processed on the relevant machine belongs. For example, a subchromosome being 563241 of machine *i* means on machine *i* the first operation to be processed belongs to job 5, second belongs to job 6, and so on.

Fitness function: The fitness of chromosome *i* in generation *K*, denoted by $C_K(i)$, is calculated by: $F_K(i) = MAX - M_K(i)$, where $F_K(i)$ is the fitness of $C_K(i)$, $M_K(i)$ is its relevant makespan and MAX is a prescribed big enough positive integer. The bigger the fitness, the shorter the makespan.

Select policy: To form the population of new generation, a set of individuals is selected from old population to reproduce itself, the select takes place in a random way but with a probability proportional to fitness. The probability of $C_K(i)$ being selected to reproduce is:

$$P_K(i) = \frac{f_K(i) - f_K}{\sum_{j=1}^{PN}[f_K(j) - f_K]} \qquad (21)$$

where $f_K = \min\{f_K(i),\ i = 1, \cdots, PN\}$ and *PN* is the size of population.

Genetic operator: The crossover operators used are Partially Mapped Crossover (PMX) and Uniform Crossover (UX); The mutation operator are Converse Mutation (CM), Right Shift Mutation (RSM) and Swap Mutation (SM). All genetic operations are limited in subchromosome to create meaningful "children".

During the processes of crossover and mutation, the starting times of operations are changed according to the changes of relevant genes. For example, table 1 shows a subchromosome and the starting times of operations to which relevant genes correspond orderly before and after crossover or mutation operation.

Status	Subchromosome	Starting times
Pre-operation	1 2 3 4 5 6	0,2,6,9,12,14
Operating	…	…
Post-operation	5 1 3 4 6 2	0,2,6,9,12,14

Table 1 STARTING TIMES CHANGES WITH GENES

Through the operating of genetic operator, the chromosomes which represent feasible solutions may be changed into chromosomes which represent non-feasible solutions. Then the obtained non-feasible solutions are processed by CSANN to solve new feasible solutions to which corresponding chromosomes form the population of the new next generation.

Stop threshold: The maximal generation (*MG*) is used.

4.2 MAIN STEPS OF HYBRID APPROACH

Step 1: Set values for *PN*, *MG*, the crossover probability P_C and the mutation probability P_M;

Step 2: Randomly create *PN* chromosomes, the initial solutions of chromosomes are deduced as follows: Assuming a subchromosome being $J_1\ J_2 \cdots J_n$, $J_i \in \{1, \cdots, n\}$ ($i \in \{1, \cdots, n\}$), J_i is the job number to which operation *i* on relevant machine belongs, then the starting times of the operations are calculated by: $S_1 = 0,\ S_{i+1} = S_i + T_i,\ i \in \{1, \cdots, n-1\}$, where S_i and T_i are the starting time and processing time of operation *i*. The obtained initial solution is used as the initial state of CSANN to solve feasible solution. The chromosomes conversed from these feasible solutions constitute the initial population, *i.e.*, the first generation.

Step 3: Calculate the fitness for each chromosome;

Step 4: Calculate the selecting probability for each chromosome and randomly select individuals according to the probability into the mate pool to reproduce;

Step 5: From the selected set of individuals select two chromosomes to generate new chromosomes by applying different crossover and mutation operators, the obtained chromosomes represent nonfeasible solutions. Use CSANN to solve feasible solutions from these nonfeasible solutions. The chromosomes conversed from obtained feasible solutions constitute the new population, *i.e.* the new generation. From the new generation randomly select an individual to be replaced by the best individual of last generation.

Step 6: If stop threshold is reached, stop the process, otherwise go to step 3.

5. SIMULATION STUDY

We use a famous 6/6/J/$C_{\max}$ problem [4] as example, which has minimal makespan of 55. The simulations are finished on a PC 586/133 under Visual C++ 5.0.

We first use CSANN only to solve the example problem to test the performance of CSANN. 100 experiments are executed. For 100 experiments the first one is executed under zero initial condition with initial starting times of all operations setting to zero, the other 99 experiments are carried out with the initial starting times of all operations valued in a random uniformed distribution between [0,100]. Of all experiments, the two parameters are valued as follows: $H = 5$, $W = 0.5$. The completion time restriction for all jobs is given in advance, which is used as the common due date for jobs in the simulations. And the release dates for all jobs are set to zero. Table 2 shows the statistics of simulation results with respect to average, minimum and maximum of obtained makespan and the program runtimes respectively. In Table 2, runtime is zero

means it is less than one second.

Completion time restriction	Initial starting times set for operations	Makespan (E)	Runtime (Sec.)
		ave/min/max	ave/min/max
200	0	76	1
200	Randomly generated	106/96/117	1/0/1
58	0	58	50
58	Randomly generated	58/58/58	48/25/97

Table 2 SIMULATION RESULTS BY CSANN ONLY

Table 2 shows the feasibility of CSANN for job-shop scheduling problems. From table 2 we can see: 1) Given zero initial solution, the CSANN can find good schedules for different complete time restriction; 2) Given appropriate complete time restriction, the CSANN can always find very good solution for most actual problems; 3) Because of the parallel processing capability of neural network, the solving speed of CSANN is very high.

Secondly we use the proposed hybrid approach to solve the same problem to test its performance. The parameters of CSANN are the same with above case: $H = 5$, $W = 0.5$. And the complete time restriction for CSANN is set to a positive integer , big enough for CSANN to solve feasible solutions, e.g. 500. The parameters of GA are as follows: PN=20, MG=200, MAX=10000, P_C =1 and P_M =0.4. Table 3 presents the simulation results as to the average, minimum and maximum of makespan and the program runtimes respectively.

Crossover mode	PMX				UX			
Mutation mode	Makespan (E)			Run time (Sec.)	Makespan (E)			Run time (Sec.)
	ave	min	max		ave	min	max	
CM	62	60	72	137	60	58	65	114
RSM	64	61	78	150	63	60	69	135
SM	63	61	74	145	62	59	75	130

Table 3 RESULTS BY HYBRID APPROACH

Fig. 3 shows a Gantt chart of a near-optimal solution with makespan being 58, obtained by the hybrid approach. From table 4 and Fig.3 we can see: the hybrid approach has good performance for job-shop scheduling problems with respect to the quality of solutions and the solving speed. Comparing between the crossover and mutation operator we can find that the UX crossover operator is better than PMX for the hybrid approach and the CM mutation operator is better than the other two mutation operators: RSM and SM.

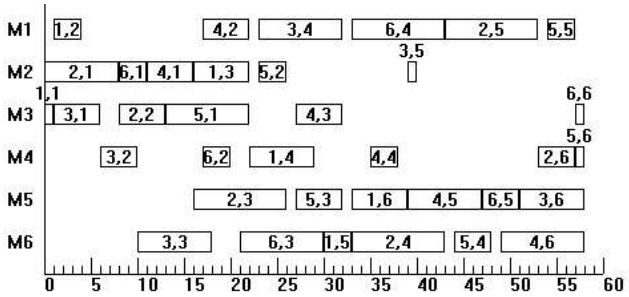

Fig.3 A NEAR-OPTIMAL SOLUTION

6. CONCLUSIONS

The proposed hybrid approach for job-shop scheduling is an idea originated from combining generic algorithm and CSANN. The adaptive property of CSANN makes it different from other constraints satisfaction networks and results in a simpler architecture of CSANN. When only CSANN is used for practical job-shop scheduling problems the quality of obtained feasible solution heavily depends on the choice of a complete time restriction.

When GA is used in a hybrid approach with CSANN, good schedules can always been obtained independent on the complete time restriction prescribed for CSANN. Simulations have shown that the proposed hybrid approach for job-shop scheduling has good performance as to the quality of solution and the speed of calculation.

Acknowledgment. This research is supported by the National Nature Science Foundation and National High -Tech Program of P. R. China.

REFERENCES

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

[2] S. French, *Sequencing and scheduling: An introduction to the mathematics of the Job-shop* (New York: Wiley, 1982)

[3] Hsiao-Lan Fang, Peter Ross and Dave Corne, A promising genetic algorithm approach to job-shop scheduling, rescheduling and open-shop scheduling problems, *Proc. 5th Int. Conf. Genetic Algorithms*, 1994, 81-100.

[4] R. Nakano and T. Yamada, Conventional genetic algorithm for job-shop scheduling, *Proc. 4th Int. Conf. on Genetic Algorithms and Their Application*, 1991.

[5] S. Y. Foo and Y. Takefuji, Neural networks for solving job-shop scheduling: Part 1. Problem representation, *Proc. IEEE IJCNN*, Ⅱ, 1988 ,275-282.

[6] D. N. Zhou, V. Charkassky, T. R. Baldwin and D. W. Hong, Scaling neural network for job-shop scheduling, *Proc. IEEE IJCNN*, New York, 3, 1989, 889-894.

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