Constraint Satisfaction Adaptive Neural Network and Heuristics Combined Approaches for Generalized Job-Shop Scheduling

Shengxiang Yang and Dingwei Wang

***Abstract*—This paper presents a constraint satisfaction adaptive neural network, together with several heuristics, to solve the generalized job-shop scheduling problem, one of NP-complete constraint satisfaction problems. The proposed neural network can be easily constructed and can adaptively adjust its weights of connections and biases of units based on the sequence and resource constraints of the job-shop scheduling problem during its processing. Several heuristics that can be combined with the neural network are also presented. In the combined approaches, the neural network is used to obtain feasible solutions, the heuristic algorithms are used to improve the performance of the neural network and the quality of the obtained solutions. Simulations have shown that the proposed neural network and its combined approaches are efficient with respect to the quality of solutions and the solving speed.**

***Index Terms*—Adaptive neural network, constraint satisfaction, generalized job-shop scheduling problem, heuristic.**

I. Introduction

PRODUCTION scheduling is the allocation of resources over time to perform a collection of tasks [1]. Of all kinds of production scheduling problems, the job-shop scheduling problem is one of the most complicated and typical. It aims to allocate m machines to perform n jobs in order to optimize certain criterion [2].

Traditionally there are three kinds of approaches for the solution of job-shop scheduling problems: priority rules, combinatorial optimization, and constraints analysis [3]. More recently intelligent knowledge-based scheduling systems have been presented [4], [5]. Since Hopfield first used a neural network to solve an optimization problem [6], Hopfield networks have been successfully applied to a variety of problems, such as the analog-to-digital conversation problem [7], the traveling salesman problem [8], the resource allocation problem [9], the linear and nonlinear programming problems [10]. However, Hopfield networks have drawbacks such as failing to converge to a valid solution, an inability to locate the global minimum and poor scaling properties due to the use of quadratic energy functions [11].

Foo and Takefuji [12] first used a neural network to solve job-shop scheduling problems. Following that, several neural-network architectures have been presented to solve job-shop scheduling problems [13]–[20]. Willems *et al.* [18], [19], first proposed a constraint satisfaction neural network for solving traditional job-shop scheduling problems with no free operations. Yu, in his Ph. D. dissertation [20], developed Willems's neural network by adding a block (called the job constraint block), which is structurally similar to the RC-block of Willems's network but its function is to deal with the problem of accomodating free operations. Additionally, Haibin introduced the gradient optimization function into its neural network for job-shop scheduling problems.

The above mentioned neural networks are basically nonadaptive networks, of which the neural units' connection weights and biases must be prescribed in advance before application of the networks to a particular problem. In this paper we propose a constraint satisfaction adaptive neural network (CSANN) for the generalized job-shop scheduling problem, accomodating free sequence operation pairs or free operations of each job. The proposed CSANN has the ability to easily map the constraints of a scheduling problem into its architecture and remove the violation of the mapped constraints during its processing and as such is based on "constraint satisfaction." Additionally CSANN has ability to adaptively adjust its connection weights and bias of neural units according to the actual constraint violations present during processing. This removes the violations in an adaptive manner. To improve the performance of CSANN for job-shop scheduling problems a mechanism of combining several heuristic algorithms with CSANN is presented. In these combined approaches CSANN is used to obtain feasible solutions and heuristic algorithms are used to improve the performance of CSANN and the quality of obtained solutions. Simulations have shown that CSANN has good performance with respect to the quality of solutions and the computing speed, especially when combined with presented heuristic algorithms, the optimal or near-optimal solutions can be found. Simulations have also shown that the proposed combined approaches can guarantee to obtain feasible solutions to realistic problems.

The organization of this paper is as follows. Section II first describes some basic concepts of job-shop scheduling, and then presents the mathematical formulation of the generalized job-shop scheduling problem. Section III presents in detail the model of CSANN including its neural unit model, its adaptive

Manuscript received December 26, 1997; revised February 11, 1999 and October 21, 1999. This work was supported by the Chinese National Natural Science Foundation under Grant 69684005 and the Chinese National High-Tech Program under Grant 863-511-9609-003, the EPSRC under Grant GR/L81468, and was done while S. Yang was pursuing the Ph.D. degree at Department of Systems Engineering, Northeastern University, Shenyang 110006, China.

S. Yang is now a Postdoctoral Research Accociate at Department of Computer Science, King's College London, University of London, U.K. (e-mail: yang@dcs.kcl.ac.uk).

D. Wang is with Department of Systems Engineering, School of Information Science and Engineering, Northeastern University, Shenyang 110006, China.

Publisher Item Identifier S 1045-9227(00)01742-2.

connections between neural units, its architecture, and its space complexity. In Section IV we describe the heuristic algorithms that can be combined with CSANN for better performace, the combined approaches for the job-shop scheduling problem are also described in this section. Section V presents the computer simulation results with several examples to show the performance of the proposed combined approaches for job-shop scheduling. Section VI is devoted to the empirical study of the computational complexity analysis of CSANN. Finally Section VII concludes this paper.

II. Description of Job-Shop Scheduling

A. Basic Concepts of Job-Shop Scheduling

Traditionally, the job-shop scheduling problem can be stated as follows [2]: given n jobs to be processed on m machines in a prescribed order under certain restrictive assumptions. The objective of job-shop scheduling is to optimally arrange the processing order and the start times of operations to optimize certain criteria. In general, there are two types of constraint for the job-shop scheduling problem. The first type of constraint states that the precedence between the operations of a job should be guaranteed, this is a *sequence constraint*. The second type of constraint is that no more than one job can be performed on a machine at the same time, this is a *resource constraint*. A job-shop scheduling problem is completely solved if the starting times of all operations are determined, and the sequence and resource constraints are not violated. Of course, the schedule obtained should also optimize certain manufacturing system criterion, such as the stocksize to be maintained, the due date reliability, the mean lead time, and the makespan (the time required to process all of the given set of jobs) [23]. Minimizing the makespan will be considered as the optimization criterion in this paper.

For a *traditional job-shop scheduling problem*, there are sequence constraints for the operations of each job [21], that is, for any two operations of a job there is a sequence constraint. In this paper we consider the *generalized job-shop scheduling problem*: there may be free sequence operation pairs or free operations for each job; there may be different number of operations for each job; there may be a release date or due date restriction for each job; and there may exist the situation that each machine can process more than one operation of a job. A free sequence operation pair of a job is a pair of two operations that have no sequence constraint. A free operation of a given job means that the operation has no sequence constraints with other operations of that particular job and can be processed before or after any other operations of the job.

Generally speaking, traditional job-shop scheduling belongs to a large class of NP-complete (nondeterministic polynomial time complete) problems. Because of the NP-complete characteristic of job-shop scheduling, it is difficult to find an optimal solution. However an optimal solution in the mathematical sense is not always required at the practical level. Thus research has concentrated on searching its near-optimal solutions using heuristic algorithms [21]. The generalized job-shop scheduling problem is much more complicate than the traditional job-shop scheduling problem and obviously belongs to NP-complete problems.

The notation system of Conway [2] will be used to represent the job-shop scheduling problem. This notation system uses four parameters of the form $n/m/A/B$ to represent a scheduling system, in which n is the number of jobs, m is the number of machines, A is the operation pattern (e.g., J means job-shop), and B is the optimization criterion (e.g., $C_{\max}$ means minimizing the maximal completion time or makespan).

B. Mathematical Formulation of Job-Shop Scheduling

To map job-shop scheduling problems onto neural networks, several pure and mixed integer programming models have been used to represent job-shop scheduling problems [14]–[17], [17], [19], and [20]. In this paper we have used the pure integer mathematical model to translate the sequence constraints, the resource constraints, the release date, and due date constraints of jobs into integer linear inequalities. This model can easily map job-shop scheduling problems onto CSANN as described in Section III.

First some notations are defined for the convenience of formulating the job-shop scheduling problem as follows: we denote $N = \{1, \cdots, n\}$ and $M = \{1, \cdots, m\}$ as the job set and the machine set, where n and m are the numbers of jobs and machines, respectively. Let n_i be the operation number of job i. O_{ikq} represents operation k of job i to be processed on machine q, S_{ikq} and T_{ikq} represent the starting time and processing time (which is known in advance) of O_{ikq}, respectively, S_{ie_iq} and T_{ie_iq} represent the starting time and processing time of the last operation of job i, respectively. Denoting 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}, and Q_i denote the set of operation pairs $[O_{ikp}, O_{ilq}]$ without precedence restriction of job i, where operation O_{ikp} and operation O_{ilq} of job i can be processed in any orders. Let R_q be the set of operations O_{ikq} that will be processed on machine q. We also assume that the starting times and the processing times of all operations are integer, and that operations cannot be interrupted once started.

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

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

subject to

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

$$S_{ilq} - S_{ikp} \geq T_{ikp} \text{ or } S_{ikp} - S_{ilq} \geq T_{ilq} \quad [O_{ikp}, O_{ilq}] \in Q_i, \qquad k, l \in \{1, \cdots, n_i\}, \quad i \in N \tag{2}$$

$$S_{jlq} - S_{ikq} \geq T_{ikq} \text{ or } S_{ikq} - S_{jlq} \geq T_{jlq} \quad O_{ikq}, O_{jlq} \in R_q, \qquad i, j \in N, \quad q \in M \tag{3}$$

$$S_{ijq} \geq r_i, \qquad i \in N, j \in \{1, \cdots, n_i\}, \quad q \in M \tag{4}$$

and

$$S_{ijq} \le d_i - T_{ijq}, \qquad i \in N, j \in \{1, \cdots, n_i\}, \quad q \in M \quad (5)$$

where (1) means that two operations with precedence restriction of a job cannot be processed at the same time and must be processed according the sequence constraint; (2), in a disjunctive type, means that two operations without sequence constraint of a job cannot overlap in time; (3), in a disjunctive type, means that a machine can only process one operation at a time, this represents the resource constraints; (4) represents the release date constraints and (5) represents the due date constraints. The cost function E is the ending time of the latest operation or operations, i.e., the maximal complete time of a given job-shop scheduling problem. Minimizing E means minimizing the makespan.

From above description and mathematical model, we can see that the problem considered is extended above the traditional $n/m/J/C_{\max}$ problem where each job passes through each machine once in a prescribed sequencing order, that is, without free operations or free sequence operation pairs. In the traditional problem there is no sequence constraint inequality of equation type (2). With the increase in the free operations or free sequence operation pairs, the number of sequence constraint inequalities of equation type (1) decreases, while the number of sequence constraint inequalities of equation type (2) increases. And to the limit, when all operations become free, the problem becomes the *open-shop* scheduling problem without sequence constraint inequalities of equation type (1).

III. Constraint Satisfaction Adaptive Neural Network

To solve the job-shop scheduling problem, the integer mathematical representation has to be mapped to CSANN. In this section CSANN will be discussed in detail with respect to its basic components of units and connections, its architecture and complexity.

A. Neural Units

Generally a neural network consists of many interconnected parallel processing elements called neural units [24]. These units compute from local information stored and transmited via connections. In general, a unit i 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 corresponding connection weights W_{ij}, together with a bias B_i. The output of summator is the net input N_i of unit 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 respectively defined as follows:

$$N_i = \sum_{j=1}^{n} (W_{ij} \times A_j) + B_i \quad (6)$$

$$A_i = f(N_i) \quad (7)$$

where W_{ij} is the connection weight from unit j to unit i. Different unit functions are realised by the use of several types of activation function, such as linear threshold function, linear-segmented function and S-shaped function [25]. In this paper two kinds of linear-segmented function A and B [see Fig. 2(a) and Fig. 2(b)] are used as the activation functions of neural units.The proposed CSANN contains three kinds of unit, based on the general neural unit. The first kind of unit are called ST-*units*, representing the starting times of all operations. Each ST-unit represents one operation of the job-shop scheduling problem, with its activation corresponding to the starting time of the particular operation. The second kind of unit, SC-*units*, represent whether the sequence constraints are violated. The third kind of unit, RC-*units*, represent whether the resource constraints are violated.

The net input of a ST-unit, e.g., ST_i, is calculated by

$$N_{ST_i}(t) = \sum_j (W_{ij} \times A_{SC_j}(t)) + \sum_k (W_{ik} \times A_{RC_k}(t)) + A_{ST_i}(t-1) \quad (8)$$

where the net input of the unit ST_i is the sum of three terms, as shown in the right side of (8). The first term represents the weighted activations of SC-units connecting with unit ST_i, which implements feedback adjustments because of sequence violations. The second term represents the weighted activations of RC-units connected with the unit ST_i, implementing feedback adjustments because of resource violations. The third term represents the previous activation, with the weight being $+1$, of the unit ST_i itself.

The activation function of ST-units is a deterministic linear-segmented function of type B [as illustrated in Fig. 2(b)] and is defined as follows:

$$A_{ST_i}(t) = \begin{cases} r_i, & N_{ST_i}(t) < r_i \\ N_{ST_i}(t), & r_i \le N_{ST_i}(t) \le d_i - T_{ST_i} \\ d_i - T_{ST_i}, & N_{ST_i}(t) > d_i - T_{ST_i} \end{cases} \quad (9)$$

where r_i and d_i are the release date and due date, respectively, 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 represented by (9) implements the release date and due date constraints described by (4) and (5).

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 sharing the same machine. The net input of a SC-unit or a RC-unit has the form of (10).

$$N_{C_i}(t) = \sum_j (W_{ij} \times A_{ST_j}(t)) + B_{C_i} \quad (10)$$

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

The activation function of a SC-unit or a RC-unit is a deterministic linear-segment function of type A [as shown in Fig. 2(a)], defined as follows:

$$A_{C_i}(t) = \begin{cases} 0, & N_{C_i}(t) \ge 0 \\ -N_{C_i}(t), & N_{C_i}(t) < 0. \end{cases} \quad (11)$$

Zero activation of a SC-unit or a RC-unit means that the corresponding sequence constraint or resource constraint is satisfied and there are no feedback adjustments from this SC-unit or RC-unit to connected ST-units. While greater than zero activation of a SC-unit or a RC-unit means that 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 the adaptive weighted connections.

B. Adaptive Connection Weights and Unit Biases

The connections of a unit transmit the activation of the unit to its connected units. The efficiency of a connection depends on the weight imposed on the connection. The received input is multiplied by this weight before it is sent to the computing unit. The connection weight can have an inhibitory effect for the computing unit when its value is negative or excitatory effect when its value is positive. The weights of connections have to be determined to achieve the desired functionality of the resulting network. Generally for constraint satisfaction neural networks, the determination of weights is executed by the designer of the neural network and the weights are determined in advance, i.e., before the network begins to solve a specific constraint satisfaction problem [15], [19], [20]. In the proposed CSANN, the connection weights and biases of neural units are adaptively valued according to the actual activations of ST-units whilst the network is running, together with the sequence and resource constraints of the specific problem.

All units of CSANN, including ST-units, SC-units, and RC-units, are connected according to the two kinds of sequence and resource constraint of a specific job-shop scheduling problem, resulting in two blocks: SC-block (sequence constraints block) and RC-block (resource constraints block). The SC-block consists of ST-units and SC-units. The RC-block consists of ST-units and RC-units. Each unit of an 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 violated (see Fig. 3). Each unit of an 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 violated (see Fig. 4). Figs. 3 and 4 show how the adaptive weights are valued. Fig. 3 illustrates an example of a SC-block unit, denoted by SCB_{ikl}, and Fig. 4 an example of a RC-block unit, denoted by RCB_{qikjl}. In Figs. 3 and 4, $I_{ST_{ikp}}$ is the initial value set for the ST-unit ST_{ikp}, responding to the initial starting time $S_{ikp}(0)$ of the operation O_{ikp}. In Fig. 3, the two ST-units ST_{ikp} and ST_{ilq} represent the 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}, respectively. The SC-unit SC_{ikl} represents whether the sequence constraint between O_{ikp} and O_{ilq} is violated, with $B_{SC_{ikl}}$ being its bias. Then at time t during the processing of network, the connection weights W_1, W_2, the feedback connection weights W_3, W_4, and the bias $B_{SC_{ikl}}$ of SCB_{ikl} are adaptively valued as shown by the following three cases.

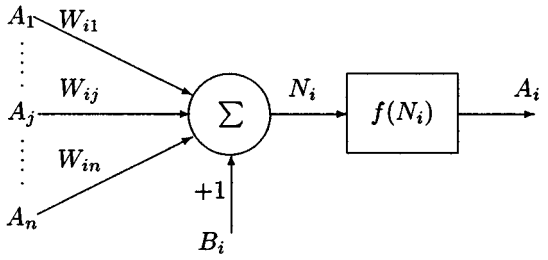

Fig. 1. General neural unit model.

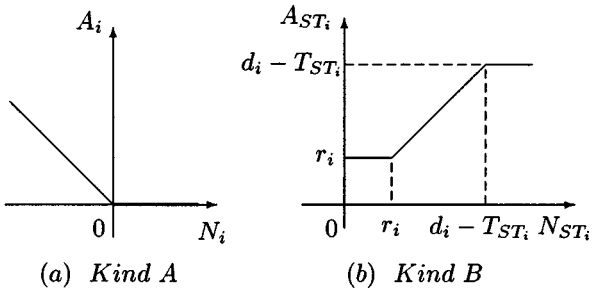

Fig. 2. Linear-segmented activation functions.

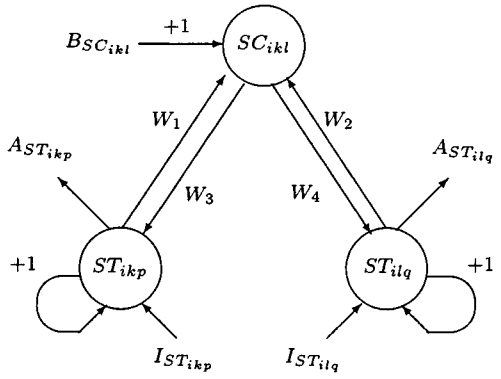

Fig. 3. A SC-block unit.

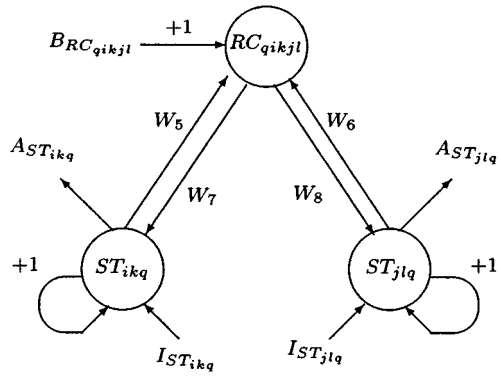

Fig. 4. A RC-block unit.

Case S1: If $[O_{ikp}, O_{ilq}] \in P_i$, the weights and bias are valued as follows:

$$W_1 = -1, \quad W_2 = 1, \quad W_3 = -W, \quad W_4 = W, \quad B_{SC_{ikl}} = -T_{ikp} \tag{12}$$

where W is a positive changeable parameter (e.g., 0.5) used for feedback adjustment (the same with following equations where W appears). In this case SCB_{ikl} represents the sequence constraint of (1). If there is no sequence constraint violation between O_{ikp} and O_{ilq}, the activation of SC_{ikl} equals zero. If there exists the sequence constraint violation, the activation of SC_{ikl} is calculated by

$$\begin{aligned} A_{SC_{ikl}}(t) &= -N_{SC_{ikl}}(t) = A_{ST_{ikp}}(t) + T_{ikp} - A_{ST_{ilq}}(t) \\ &= S_{ikp}(t) + T_{ikp} - S_{ilq}(t) \end{aligned} \tag{13}$$

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

$$\begin{aligned} A_{ST_{ikp}}(t+1) &= S_{ikp}(t+1) \\ &= A_{ST_{ikp}}(t) + W_3 \times A_{SC_{ikl}}(t) \\ &= S_{ikp}(t) - W \times A_{SC_{ikl}}(t) \end{aligned} \tag{14}$$

$$\begin{aligned} A_{ST_{ilq}}(t+1) &= S_{ilq}(t+1) \\ &= A_{ST_{ilq}}(t) + W_4 \times A_{SC_{ikl}}(t) \\ &= S_{ilq}(t) + W \times A_{SC_{ikl}}(t). \end{aligned} \tag{15}$$

From the above equations we can see that the effect of unit SC_{ikl} to unit ST_{ikp} is inhibitory, which leads the starting time of operation O_{ikp} being put back on the time axis. While the effect of SC_{ikl} to ST_{ilq} is excitatory, putting forward S_{ilq}. Thus the sequence violation between O_{ikp} and O_{ilq} can be removed.

Case S2: If $[O_{ikp}, O_{ilq}] \in Q_i$ and $S_{ikp}(t) \leq S_{ilq}(t)$, the adaptive weights and bias are valued the same with **Case S1**, using (12). In this case SCB_{ikl} represents the sequence constraint of first disjunctive equation of (2). If there exists the violation, the activation of SC_{ikl} and its feedback adjustments are calculated the same way as with **Case S1**, using (13)–(15).

Case S3: If $[O_{ikp}, O_{ilq}] \in Q_i$ and $S_{ikp}(t) \geq S_{ilq}(t)$, the weights and bias are adaptively valued by

$$\begin{aligned} &W_1 = 1, \quad W_2 = -1, \quad W_3 = W \\ &W_4 = -W, \quad B_{SC_{ikl}} = -T_{ilq}. \end{aligned} \tag{16}$$

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

$$\begin{aligned} A_{SC_{ikl}}(t) &= A_{ST_{ilq}}(t) + T_{ilq} - A_{ST_{ikp}}(t) \\ &= S_{ilq}(t) + T_{ilq} - S_{ikp}(t) \end{aligned} \tag{17}$$

$$\begin{aligned} A_{ST_{ikp}}(t+1) &= A_{ST_{ikp}}(t) + W_3 \times A_{SC_{ikl}}(t) \\ &= S_{ikp}(t) + W \times A_{SC_{ikl}}(t) \end{aligned} \tag{18}$$

$$\begin{aligned} A_{ST_{ilq}}(t+1) &= A_{ST_{ilq}}(t) + W_4 \times A_{SC_{ikl}}(t) \\ &= S_{ilq}(t) - W \times A_{SC_{ikl}}(t). \end{aligned} \tag{19}$$

Similarly in Fig. 4, RCB_{qikjl} represents the resource constraint between O_{ikq} and O_{jlq} on machine q. At time t during the processing of the network, the adaptive weights and bias are valued as shown by the following two cases.

Case R1: For O_{ikq} and $O_{jlq} \in R_q$, if $S_{ikq}(t) \leq S_{jlq}(t)$, we get

$$\begin{aligned} &W_5 = -1, \quad W_6 = 1, \quad W_7 = -W \\ &W_8 = W, \quad B_{RC_{qikjl}} = -T_{ikq}. \end{aligned} \tag{20}$$

In this case RCB_{qikjl} represents a sequence constraint described by the first disjunctive equation of (3). If there exists a violation, the activation of RC_{qikjl} and the feedback adjustments from RC_{qikjl} to ST_{ikq} and ST_{jlq} are calculated by

$$\begin{aligned} A_{RC_{qikjl}}(t) &= A_{ST_{ikq}}(t) + T_{ikq} - A_{ST_{jlq}}(t) \\ &= S_{ikq}(t) + T_{ikq} - S_{jlq}(t) \end{aligned} \tag{21}$$

$$\begin{aligned} A_{ST_{ikq}}(t+1) &= A_{ST_{ikq}}(t) + W_7 \times A_{RC_{qikjl}}(t) \\ &= S_{ikq}(t) - W \times A_{RC_{qikjl}}(t) \end{aligned} \tag{22}$$

$$\begin{aligned} A_{ST_{jlq}}(t+1) &= A_{ST_{jlq}}(t) + W_8 \times A_{RC_{qikjl}}(t) \\ &= S_{jlq}(t) + W \times A_{RC_{qikjl}}(t). \end{aligned} \tag{23}$$

Case R2: For O_{ikq} and $O_{jlq} \in R_q$, if $S_{ikq}(t) \geq S_{jlq}(t)$, we get

$$\begin{aligned} &W_5 = 1, \quad W_6 = +1, \quad W_7 = W \\ &W_8 = -W, \quad B_{RC_{qikjl}} = -T_{jlq}. \end{aligned} \tag{24}$$

In this case RCB_{qikjl} represents a sequence constraint described by the second disjunctive equation of (3). If there exists a violation, the activation of RC_{qikjl} and the feedback adjustments are calculated by

$$\begin{aligned} A_{RC_{qikjl}}(t) &= A_{ST_{jlq}}(t) + T_{jlq} - A_{ST_{ikq}}(t) \\ &= S_{jlq}(t) + T_{jlq} - S_{ikq}(t) \end{aligned} \tag{25}$$

$$\begin{aligned} A_{ST_{ikq}}(t+1) &= A_{ST_{ikq}}(t) + W_7 \times A_{RC_{qikjl}}(t) \\ &= S_{ikq}(t) + W \times A_{RC_{qikjl}}(t) \end{aligned} \tag{26}$$

$$\begin{aligned} A_{ST_{jlq}}(t+1) &= A_{ST_{jlq}}(t) + W_8 \times A_{RC_{qikjl}}(t) \\ &= S_{jlq}(t) - W \times A_{RC_{qikjl}}(t). \end{aligned} \tag{27}$$

C. Architecture of CSANN

To sum up, the architecture of the network proposed in this paper is simpler than those of three-layer networks proposed by Willems *et al.* [18], [19] and Yu [20]. The architecture of the proposed network consists of two layers. The bottom layer consists only of ST-units, corresponding to the starting times of all operations. The top layer contains both SC-units and RC-units, which represent sequence and resource constraints, respectively, and provide feedback information to adjust ST-units for sequence and resource constraints satisfaction through SC-block and RC-block respectively.

To solve a specific job-shop scheduling problem, CSANN can be built up as follows: first compute the number of ST-units are according to the specific problem (the number is given by $\sum_{i\in N} n_i$), then build up the three sets of P_i, Q_i and R_q according to the actual sequence and resource constraints, finally form the SC-block and RC-block, resulting in the problem-specific neural network.

D. Network Complexity Analysis

As previously mentioned in Section II, there are two limit cases of the generalized job-shop scheduling problem described by the mathematical model: the traditional job-shop scheduling problem and the open-shop scheduling problem. Correspondingly, there are also two limits for the network complexity of CSANN.

For a traditional $n/m/J/C_{\max}$ problem, we assume that $n_i = m$ for all $i \in N$ and each job passes through each machine once in a prescribed sequencing order. There are $n(m-1)$ sequence constraint inequalities of equation type (1), which requires $n(m-1)$ SC-units; there are $mn(n-1)$ resource constraint inequalities of equation type (3), which requires $mn(n-1)$ RC-units; there are mn ST-units representing the starting times of all operations. Thus the total number of neural units is $n(mn+m-1)$. Because each SC-unit or RC-unit has two incoming connections from, and two feedback connections to two ST-units, the number of interconnections in SC-block is $4n(m-1)$, and the number of interconnections in RC-block is $4mn(n-1)$. Hence the total number of interconnections in the CSANN network is $4n(mn-1)$.

For an open-shop scheduling problem, we assume that $n_i = m$ for all $i \in N$ and each job passes through each machine once. There are $n \times C_m^2 = mn(m-1)$ SC-units representing the sequence constraint inequalities of equation type (2), $mn(n-1)$ RC-units representing the resource constraint inequalities of equation type (3), mn ST-units representing the starting times of all operations, resulting in a total number of $mn(m+n-1)$ neural units. Similarly, the number of interconnections in SC-block is $4mn(m-1)$, and the number of interconnections in RC-block is $4mn(n-1)$. Hence the total number of interconnections in the CSANN network is $4mn(m+n-2)$.

For a generalized job-shop scheduling problem, we also assume that $n_i = m$ for all $i \in N$ and each job passes through each machine once, the number of SC-units is between $n(m-1)$ and $mn(m-1)$ and the total number of the neural units of the resulting CSANN is between $n(mn+m-1)$ and $mn(m+n-1)$. Obviously, the neural unit complexity of CSANN is $O(mn^2)$. Similarly, the number of interconnections in SC-block is between $4n(m-1)$ and $4mn(m-1)$, the number of interconnections in RC-block is $4mn(n-1)$. The total number of interconnections in the CSANN network is between $4n(mn-1)$ and $4mn(m+n-2)$. Hence the connection complexity of CSANN is also $O(mn^2)$.

E. Mechanisms of Running CSANN

There are three mechanisms of running CSANN. In the first mechanism, during each iteration cycle of calculating all units, the activation of units is calculated in a fixed order of first calculating each ST-unit, then calculating each SC-unit, and finally calculating each RC-unit. This simulation mechanism results in a deterministic unique schedule. That is, under the same initial conditions of ST-units, the network can converge to the unique stable state responding to the unique solution. This is an asynchronous processing mode.

The second mechanism calculates the activation of units in a random order during each iteration cycle, which results in nondeterministic schedules. That is, under the same initial conditions of ST-units, the network always converges to nondeterministic stable state, resulting in a feasible but not the same solution. This kind of mode is also asynchronous.

In the third mechanism, the activation of units is calculated in a synchronous parallel manner. During the simulation processing of network, the activation of all units is calculated in a random or fixed order, but the newly calculated activation of a unit is not sent immediately to its connected units but stored until all units have finished their calculations and stored their activations. In the next calculation cycle the activation of a unit is calculated using the stored activations of the connected units.

IV. Combined Approaches for Job-Shop Scheduling

In this section the approaches of combining CSANN and heuristic algorithms for job-shop scheduling problems are described in detail.

A. Structure of the Scheduling Approaches

The combined approaches for job-shop scheduling consist of two parts: CSANN and several heuristics presented latter in this section. In the combined scheduling approaches, the heuristic algorithms can be used individually or all together with CSANN. CSANN is used to remove the violations for sequence and resource constraints resulting in the generation of feasible solutions to the specific problem, and the heuristics are used to improve the performance of the CSANN or the quality of the feasible solutions obtained by CSANN. Fig. 5 shows the maximal structure of the combined approach where all heuristics are used. In Fig. 5, Alg. is the abbreviation of "algorithm."

B. Heuristic Algorithms

As shown in Fig. 5, in the combined approach for the job-shop scheduling, three relative heuristic algorithms that can be used individually or all together with CSANN are proposed.

Algorithm 1: Exchange the orders of two near operations by exchanging their starting times. There are two cases when this algorithm works. The first case is when the operation pairs are of the same job. The second case is when the two operations are of different jobs sharing the same machine.

For the first case, assume the operation pairs be $[O_{ikp}, O_{ilq}]$. At time t during the processing of CSANN, if $[O_{ikp}, O_{ilq}] \in P_i$ and $S_{ikp}(t) > S_{ilq}(t)$ or $[O_{ikp}, O_{ilq}] \in Q_i$ and $G_{ikl}(t) \geq G$, the following two equations begin to work:

$$A_{ST_{ikp}}(t+1) = S_{ikp}(t+1) = A_{ST_{ilq}}(t) = S_{ilq}(t) \quad (28)$$

$$A_{ST_{ilq}}(t+1) = S_{ilq}(t+1) = A_{ST_{ikp}}(t) = S_{ikp}(t) \quad (29)$$

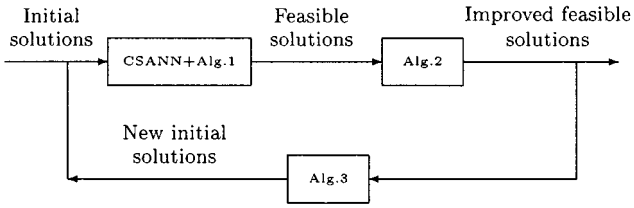

Fig. 5. The maximal combined approach structure.

where variable $G_{ikl}(t)$ sums up the times that, at time t during the processing of CSANN, operation pairs $[O_{ikp}, O_{ilq}] \in Q_i$ have their starting times S_{ikp} and S_{ilq} continuously changed with the same adjusting effect ever since the previous zero-reset because of sequence conflict. G, a prescribed positive integer, is used as the judging condition for algorithm 1 to be entriggered. That is, when $G_{ikl}(t)$ reaches G the algorithm works and resets $G_{ikl}(t)$ to zero meanwhile.

For the second case, assume $O_{ikq}, O_{jlq} \in R_q$. At time t during the processing of CSANN, if $H_{qikjl}(t) \geq H$, the following two equations begin to work:

$$A_{ST_{ikq}}(t+1) = S_{ikq}(t+1) = A_{ST_{jlq}}(t) = S_{jlq}(t) \quad (30)$$

$$A_{ST_{jlq}}(t+1) = S_{jlq}(t+1) = A_{ST_{ikq}}(t) = S_{ikq}(t) \quad (31)$$

where variable $H_{qikjl}(t)$ is the summed times that operation pairs O_{ikq} and O_{jlq} have their starting times continuously changed with 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 H is a prescribed positive integer with the same functionality as G.

Algorithm 1 can be used together with CSANN to guarantee the generation of feasible solutions. During the processing of CSANN there may appear the phenomenon of so-called "dead lock" which can result in no feasible solution. This phenomenon results from the conflicts of feedback adjustments themselves while removing sequence and resource constraint violations. For example, assume $[O_{ikp}, O_{ilq}] \in P_i$ or $[O_{ikp}, O_{ilq}] \in Q_i$, 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} 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 the two kinds of adjustments which result in "dead lock." "Dead lock" leads to the nonconvergence of CSANN to its stable station, which corresponds to the feasible solution of specific job-shop scheduling problem. By using algorithm 1, when the phenomenon of "dead lock" happens and S_{ilq} has been continuously put back H times because of resource violation between O_{ilq} and O_{jmq}, that is, at time t $H_{qikjl}(t)$ reaches H, 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 the feasible solution is guaranteed.

Algorithm 2: Compact away the idle times. That is to eliminate the time segments in the feasible solution obtained by CSANN, during which all machines are idle. Thus an improved feasible solution with shorter makespan can be obtained.

Let N_{IT} be the number of idle time segments existing in the feasible solution, and IT_i $(i \in \{1, \cdots, N_{IT}\})$ be the time length of ith idle time segment with the starting point and ending point being IS_i and IE_i, respectively, then algorithm 2 is presented as follows:

$$S'_{ikp} = S_{ikp} - \sum_{j=1}^{l-1} IT_j, \qquad \text{if } IE_{l-1} < S_{ikp} \leq IS_l \quad (32)$$

where $l \in \{1, \cdots, N_{IT}\}$, S_{ikp} is the starting time of O_{ikp} in the old feasible solution, and S'_{ikp} is the new starting time of O_{ikp} in the improved feasible solution with all idle time segments within S_{ikp} compacted away.

Algorithm 3: Shorten the starting times of those operations with maximal complete time in the obtained feasible solution. That is, for the latest operations of all jobs, if the equation of $S_{ie_iq}(t) + T_{ie_iq} = \max_{j \in N} (S_{je_jq} + T_{je_jq})$ holds, (33) will work

$$S'_{ie_iq} = S_{ie_iq} - \delta, i \in N \quad (33)$$

where S'_{ie_iq} is the new starting time, δ is a positive changeable parameter.

Algorithm 3 is originated from literature [20] and is used to obtain a new initial solution, which can be reused by CSANN in order to solve better solutions. In fact, the cost function E of the job-shop scheduling model presented in Section II can be seen as the energy function of CSANN. The functionality of (33) is to decrease the energy of the network and direct it toward the optimal solution.

C. Description of the Combined Approaches

The basic steps of the combined approaches for solving the job-shop scheduling problem are shown as follows:

- Step 1) Build up CSANN model, set the values for parameters G, H, W and δ, and prescribe an expected makespan;
- Step 2) Randomly initialize or specify by hand the initial starting time $S_{ikp}(0)$ for each operation O_{ikp} $(i \in N, k \in \{1, \cdots, n_i\})$, and let $S_{ikp}(0)$ be 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 $A_{SC_{ikl}}(t)$ with (13) or (17). $A_{SC_{ikl}}(t) \neq 0$ means the dissatisfaction of sequence constraint corresponding to (1) or (2), then adjust activations of relative ST-units with (14) and (15) or (18) and (19), or with (28) and (29) if algorithm 1 is comdined in the approach and its conditions are satisfied;
- Step 4) Run each RC-unit RC_{qikjl} of RC-block, calculate its activation $A_{RC_{qikjl}}(t)$ with (21) or (25). $A_{RC_{qikjl}}(t) \neq 0$ means the dissatisfaction of resource constraint corresponding to (3). In this case adjust $S_{ikq}(t+1)$ and $S_{jlq}(t+1)$ with (22), (23) or (26), (27) according to the actual situation or with (30), (31) if algorithm 1 is combined and its conditions are satisfied;
- Step 5) Repeat Steps 3) and 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. If algorithms 2 and 3 are not used, stop the program now;

Step 6) If algorithm 2 is used, run algorithm 2 in order to obtain an improved feasible solution. If algorithm 3 is not used, stop the program now;

Step 7) If algorithm 3 is used, judge whether the prescribed stop criterion is achieved. If the stop criterion is achieved, stop the program; otherwise, take the makespan of the newly obtained solution as new expected makespan, use algorithm 3 to obtain a new initial solution and return Step 3).

In Steps 1) and 7), the expected makespan is the makespan the scheduler want to achieve. For a scheduling problem without due date constraints, the expected makespan can be used as the common due date of all jobs. For a scheduling problem with common or different due dates, the expected makespan can be set to the biggest due date of all jobs. In Step 7), the stop criterion used is the continuous run times, e.g., ten times, with the makespans of feasible solutions obtained keeping the same with ever obtained shortest makespan of feasible solutions.

V. Simulation Study

A. Simulation Examples

Four job-shop scheduling problems of $3/2/J/C_{\max}$, $5/3/J/C_{\max}$, $6/6/J/C_{\max}$, and $10/10/J/C_{\max}$ are presented as simulation examples.

Example 1: Table I presents the original data of a $3/2/J/C_{\max}$ generalized job-shop scheduling problem, where there are free operations and different number of operations for different jobs. In Table I, the previous or next operation being equal to zero represents the case that the responding operation has no previous or next operation sequence constraint. If an operation has neither previous operation nor next operation sequence constraint, it is called a free operation, e.g., the two operations of job 2. Of job 3 the operation 3 has two previous operations 1 and 2, operation 1 and operation 2 are free sequence operation pair. Each of job 1 and job 3 has three operations, while job 2 has two operations.

Example 2: Table II presents a $5/3/J/C_{\max}$ problem, which is also a generalized job-shop scheduling problem with different due dates and different number of operations for jobs. In Table II, (m, t) means that the relevant operation of some job will be processed on machine m with its processing time being t (the same with following Tables III and IV). The sequence constraints of all jobs are the same: in order from operation 1 to operation 3. In this example, job 3 has only two operations, job 4 has its first operation and third operation to be processed on machine 3. The due dates for the five jobs are 23, 25, 15, 30, and 30, respectively.

Example 3: Table III presents a traditional $6/6/J/C_{\max}$ problem from literature [26]. The sequence constraints of all jobs are the same: in order from operations 1 to 6. This example has the optimum (i.e., minimal makespan) of 55, which is already known.

Example 4: Table IV presents a traditional $10/10/J/C_{\max}$ problem, with its data measured from the feasible schedule presented by Zhou *et al.* [16]. The sequence constraints of all jobs are the same: in order from operation 1 to operation 10. The makespan of the feasible schedule given by Zhou *et al.* [16] is 98.

TABLE I
ORIGINAL DATA OF EXAMPLE 1

Job No.	1			2		3		
Operation No.	1	2	3	1	2	1	2	3
Machine No.	1	1	2	1	2	2	1	2
Previous Operation	0	1	2	0	0	0	0	1,2
Next Operation	2	3	0	0	0	3	3	0
Processing Time	3	5	6	8	4	9	2	7

TABLE II
ORIGINAL DATA OF EXAMPLE 2

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

TABLE III
ORIGINAL DATA OF EXAMPLE 3

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

B. Simulation Environment

Because of the adaptive property, it is very suitable to realize the proposed CSANN in software. Because of the object-oriented characteristics of units, it is relatively easy to simulate the CSANN with an object-oriented developing language. The simulation of CSANN was implemented on an Intel 586 PC running at 133 MHz. The development of CSANN was undertaking using Microsoft Visual C++ 5.0 development environment. The first task was to build three classes: class CSTunit, class CSCunit and class CRCunit according to the characteristics of the summation and activation functions of ST-units, SC-units and RC-units, then build up the class CNetwork as their friend class. While building these four classes, the sequence and resource constraints of a specific problem are considered. Thus the problem-specific CSANN can be built up.

In order to determine the performance of CSANN and compare between the proposed heuristic algorithms, in our simulations the deterministic fixed-order mechanism of running CSANN is used, under which the proposed network converges to an unique solution from an initial solution.

For the simulations, CSANN and proposed algorithms were used with four combination methods. Method 1 is CSANN

TABLE IV
ORIGINAL DATA OF EXAMPLE 4

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

alone. Method 2 is the combination of CSANN and Alg.1. Method 3 is the combination of CSANN, Alg.1 and Alg.2. Method 4 is the combination of CSANN, Alg.1, Alg.2 and Alg.3. Of all simulations, the four parameters are valued as follows: $G = H = \delta = 5, W = 0.5$. In all the figures and tables given subsequently in this paper, the program run time is rounded up to the next smaller integer value (in seconds). And in all the Gantts the blocks represent individual operations. The length of a block equals the processing time of relative operation, and (i, j) represents operation j of job i.

C. Simulation Results and Analyses

For **Example 1**, different methods were used to solve the problem from the same randomly initialized solution with the expected makespan prescribed to be 120. Fig. 6 shows the feasible solution in the mode of Gantt chart, obtained by Method 2. The obtained solution has six idle time segments that can be compacted away. Fig. 7 shows the solution obtained by Method 3. From Fig. 7, we can see, with algorithm 2 used, the idle times are compacted away, resulting in a makespan of 43 which is much shorter than that of 102 in Fig. 6. Fig. 8 presents a solution obtained by Method 4. From Fig. 8 we can see, with algorithm 3 used, the optimal solution is obtained with makespan of 26.

For **Example 2**, we use CSANN alone to solve from a random initial solution. Fig. 9 shows the simulation result. Obviously, Fig. 9 is an optimal solution with the due dates of all jobs satisfied.

From Figs. 7–9, we can see that the approaches are efficient for the generalized job-shop scheduling problem.

For **Example 3**, in order to show the performance of the proposed approaches, different methods were first used to solve from zero initial solution under different expected makespans. Zero initial solution means that the initial starting times of all operations is set to zero. The simulation results are shown in Table V. From Table V it can be seen that searching from zero initial solutions, CSANN can always find good schedules by different methods with all expected makespans in a short time.

Second, the use of different methods to solve **Example 3** from randomly initial solutions under different expected makespans: 300 (equivalent to $+\infty$; for **Example 3**), 100 (quite loose value) and 58 (near-optimal value) was investigated. A randomly initial solution is that the initial starting times of all operations are valued in a random and uniformed distribution in the region of [0,100]. With each method 100 experiments were executed. For each experiment the expected makespan was used as the common due date for all jobs. For all experiments the release dates for all jobs were set to zero. The statistics of the simulation results with respect to average, minimum and maximum

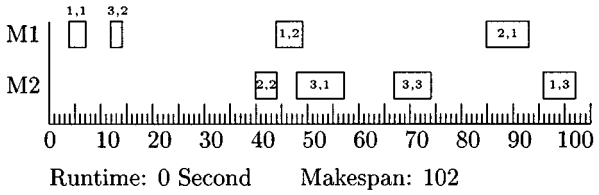

Fig. 6. A solution of **Example 1** by Method 1.

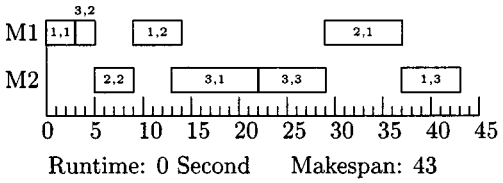

Fig. 7. A solution of **Example 1** by Method 3.

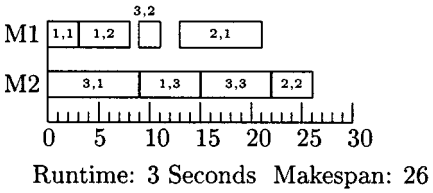

Fig. 8. A solution of **Example 1** by Method 4.

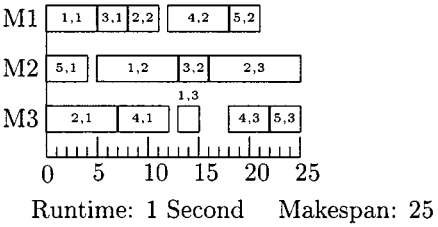

Fig. 9. A solution of **Example 2** by CSANN.

TABLE V
SIMULATION RRESULTS OF EXAMPLE 3 WITH ZERO INITIAL SOLUTIONS

Solving method	Expected makespan	Makespan (E)	Runtime (Seconds)
Method 1	$+\infty$	71	0
Method 1	100	71	0
Method 1	58	58	38
Method 2	$+\infty$	71	0
Method 2	100	71	0
Method 2	58	58	34
Method 3	$+\infty$	71	0
Method 3	100	71	0
Method 3	58	58	35
Method 4	$+\infty$	56	97

of obtained makespans, the program runtimes and the times of not converged in 100 experiments, respectively, are shown in Table VI. Fig. 10 shows the running result from a randomly initialized solution by Method 1 with the expected makespan prescribed to be 58. Fig. 11 is the optimal solution obtained from a randomly initialized solution by Method 4 with the expected makespan prescribed to 300.

From Table VI, Figs. 10 and 11, we can see that when searching from randomly initial solutions, we can get the following points.

1) By Method 1 (CSANN alone), the quality of obtained solutions heavily depends on the expected makespan. When the expected makespan is suitably prescribed, near-optimal or optimal solutions can be found.
2) By Method 1, there may appear the phenomenon of "dead lock" with the percentage of about 8%.
3) While combined with algorithm 1, CSANN can always converge to feasible solutions.
4) By Method 4, near-optimal or optimal solutions can always be found, independent on the expected makespan and initial conditions.
5) The solving speed of the proposed approaches for job-shop scheduling problems is high.

Additionally, from Figs. 6 and 7 and Tables V and VI, we can see that while the expected makespan is quite loose and the initial starting times of operations were randomly generated from a quite wide time region (e.g., [0, 100] is a quite wide region for **Example 1**), algorithm 2 can be of great effect, but while the expected makespan is quite tight or the initial starting times of operations is randomly generated from a quite tight time region (e.g., [0, 100] is a quite tight region for **Example 3**), algorithm 2 can be of little effect.

For **Example 4**, to compare our proposed approaches with other neural networks, we first use Method 1 (only CSANN) to solve it from a randomly initial solution with the expected makespan set to 100, which is near the makespan given in literature [16]. The simulation result is given in Fig. 12. Second we use Method 4 to solve **Example 4** from a randomly initial solution with the initial expected makespan set to 1000, which is much greater than the sum of processing times of all operations and equivalent to $+\infty$. The simulation result is shown in Fig. 13. From Fig. 12, we can see that CSANN obtained a quite

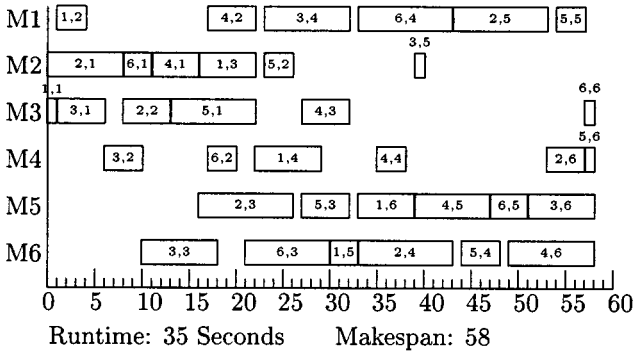

Fig. 10. A near-optimal solution of **Example 3** by Method 1.

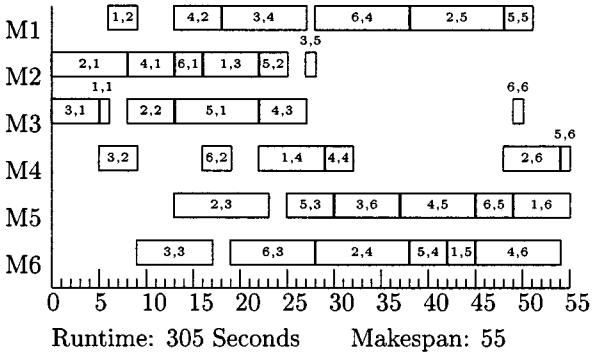

Fig. 11. An optimal solution of **Example 3** by Method 4.

good result, based on the comparison with the total processing time of the longest job. The total processing time of the longest job, i.e., job 9, is 68. From Fig. 13, we can see that the makespan of the obtained solution of **Example 4** is 95, which is better than the schedule result given in literature [16].

Simulations of above four examples proved the efficient performance of proposed CSANN and its combined approaches with several algorithms for job-shop scheduling problems as to good solutions and high solving speed.

VI. COMPUTATIONAL COMPLEXITY ANALYSIS

As mentioned in Section III, there are three mechanisms of running CSANN. The first mechanism was used in the simulations. Similarly in this section the computational complexity of CSANN was invetigated with this mechanism. Computational complexity of CSANN consists of two factors. One is the calculating times that CSANN requires during each iteration cycle. The other one is the total number of iterations that CSANN needs to obtain a feasible solution.

We first discuss the computational complexity of each iteration cycle. For the convenience of discussions and without the lose of generality, we take as the analysis example a traditional $n/m/J/C_{\max}$ problem where $n_i = m$ for all $i \in N$ and each job passes through each machine in a prescribed sequencing order. As described in Section III, there are mn ST-units, $n(m-1)$ SC-units and $mn(n-1)$ RC-units. We assume that in the worst case, in which during an iteration cycle for each SC-unit or RC-unit there are constraint violation and feedback adjustments. In this case for each iteration cycle, first

TABLE VI
SIMULATION RESULTS OF EXAMPLE 3 WITH RANDOMLY INITIAL SOLUTIONS

Solving method	Expected makespan	Makespan (Ave/Min/Max)	Runtime (Seconds) (Ave/Min/Max)	Times of not converged
Method 1	300	106/96/117	1/0/1	7
Method 1	100	94/88/100	2/0/6	8
Method 1	58	58/58/58	49/19/85	10
Method 2	300	105/95/118	1/0/1	0
Method 2	100	93/86/100	1/0/5	0
Method 2	58	58/58/58	45/17/80	0
Method 3	300	68/61/84	2/0/7	0
Method 3	100	93/85/100	2/1/8	0
Method 3	58	58/58/58	48/25/95	0
Method 4	300	60/55/67	145/91/349	0

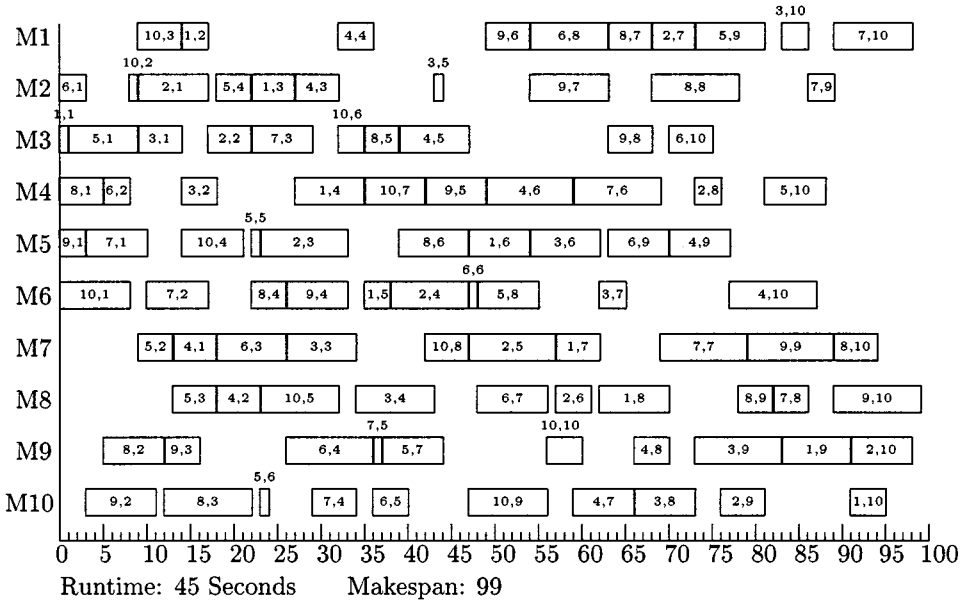

Fig. 12. A feasible solution of **Example 4** by CSANN.

mn ST-units are calculated, then $n(m-1)$ SC-units are calculated and $2n(m-1)$ feedback adjustments are calculated, finally $mn(n-1)$ RC-units and $2mn(n-1)$ feedback adjustments are calculated. So for each iteration, the total number of calculations is $n(3mn+m-3)$, which is on the order of $O(mn^2)$.

Second, the total number of iterations needed by CSANN to obtain a feasible solution is discussed. Usually the total number of iterations varies with the problem size (i. e., the total operations in our discussions) and the parameter value of CSANN. In order to test the relation between the total number of iterations and the problem size, we simulated under the same parameter value $(W = 0.5)$ with randomly generated eight traditional job-shop scheduling problems: from $3/3/J/C_{\max}$ to $10/10/J/C_{\max}$. For all these problems, each job passes through each machine once in the order of from the first operation to the last operation. In all the simulations the expected makespans were set to be $+\infty$ and the solution starts from random initial solutions. The simulation results are shown in Table VII. Table VII shows that the relation between the iteration times and the problem size is approximately linear with the ratio of the total iteration number to the problem size being about five. That is, the total number of iterations is on the order of $O(mn)$ to the problem size.

The total computational complexity of CSANN is the product of the number of iterations and the complexity per iteration, which is approximately $O(m^2n^3)$.

VII. CONCLUSIONS

The proposed approaches for job-shop scheduling are originated from the idea of combining CSANN and several heuristic algorithms. CSANN has the property of easily mapping the sequence and resource constraints of specific job-shop scheduling problem onto its architecture and removing the violations of these constraints during its processing to obtain feasible solutions. The adaptive property of CSANN makes it different from other constraint satisfaction neural networks in a simpler architecture.

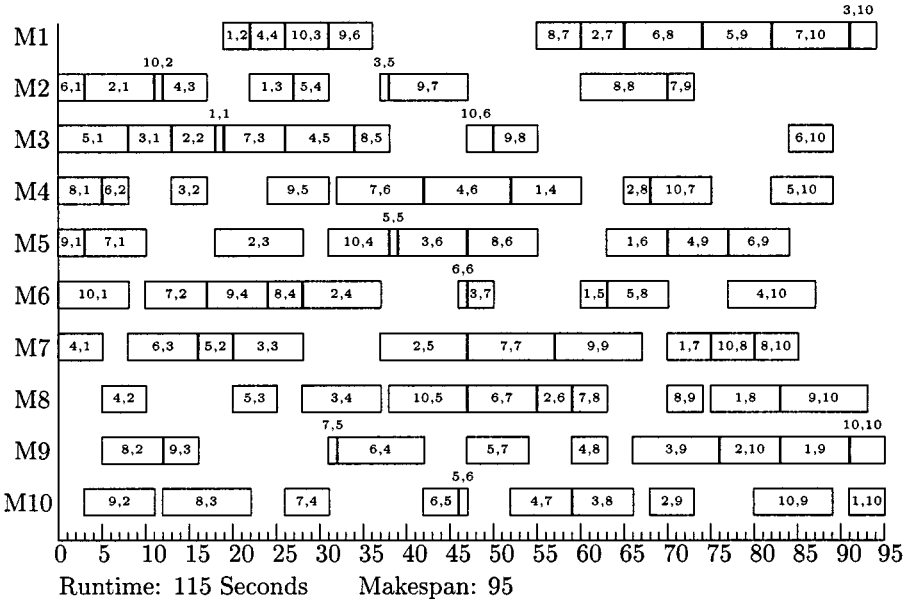

Fig. 13. A solution of **Example 4** by Method 4.

TABLE VII
SIMULATION RESULTS OF COMPUTATIONAL COMPLEXITY

Simulation Problems	Number of Operations	Runtime (Seconds)	Number of Iterations
$3/3/J/C_{max}$	9	0	37
$4/4/J/C_{max}$	16	0	64
$5/5/J/C_{max}$	25	0	134
$6/6/J/C_{max}$	36	1	161
$7/7/J/C_{max}$	49	1	213
$8/8/J/C_{max}$	64	1	334
$9/9/J/C_{max}$	81	2	412
$10/10/J/C_{max}$	100	3	584

When only CSANN is used for job-shop scheduling problems, the quality of feasible solutions obtained somewhat depends on the choice of an expected makespan which may be the scheduler's desired objective. When the expected makespan is suitably chosen, the desired objective can always be achieved. But when the specification of the expected makespan is too loose the feasible solution searched may be not good enough, and when too tight or shorter than the optimum, the feasible solution cannot be obtained. Meanwhile there may appear the phenomenon of nonconvergence among many runs.

To improve the performance of CSANN and the quality of solutions searched, we can combine CSANN with the proposed heuristics. While combined with these algorithms, CSANN can always find good schedules (including near-optimal and optimal solutions) with the expected makespan chosen quite loose or even equivalent to $+\infty$.

For practical scheduling problems we can apply the following strategy: first use only CSANN to obtain a feasible solution from the zero initial solution, then use the obtained makespan as the expected makespan to run Method 4 from randomly initial solutions several times, finally take the obtained best solution as the practical schedule.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their helpful comments and suggestions that contributed to improve the quality of this paper.

REFERENCES

[1] K. R. Baker, *Introduction to Sequence and Scheduling*. New York: Wiley, 1974.
[2] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*. Reading, MA: Addison-Wesley, 1967.
[3] D. Dubois, H. Fargier, and H. Prade, "Fuzzy constraints in job-shop scheduling," *J. Intell. Manufacturing*, vol. 6, pp. 215–234, 1995.
[4] P. V. Hentenryck, *Constraint Satisfaction and Logic Programming*. Cambridge, MA: MIT Press, 1989.
[5] M. S. Fox and M. Zweben, *Knowledge-Based Scheduling*. San Manteo, CA: Morgan Kaufmann, 1993.
[6] J. J. Hopfield and D. W. Tank, "'Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.
[7] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, single decision circuit and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. 33, pp. 533–541, 1986.
[8] J. H. Park and H. Jeong, "Solving the TSP using an effective Hopfield network," in *Proc. IEEE Int. Conf. Neural Networks*, Paris, France, 1990.
[9] G. A. Tagliarini and E. W. Page, "Solving constraints satisfaction problems with neural networks," in *Proc. IEEE 1st IJCNN*, vol. III, 1987, pp. 741–747.
[10] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 554–562, 1988.
[11] G. V. Wilson and G. S. Pawley, "On the stability of the TSP algorithm of Hopfield and Tank," *Biol. Cybern.*, vol. 58, pp. 63–70, 1988.
[12] S. Y. Foo and Y. Takefuji, "Neural networks for solving job-shop scheduling: Part 1. Problem representation," in *Proc. IEEE IJCNN*, vol. II, San Diego, CA, 1988, pp. 275–282.
[13] ——, "Stochastic neural networks for solving job-shop scheduling: Part 2. Architecture and simulations," in *Proc. IEEE IJCNN*, vol. II, San Diego, CA, 1988, pp. 283–290.

[14] ——, "Integer linear programming neural networks for job-shop scheduling," in *Proc. IEEE IJCNN*, vol. II, San Diego, CA, 1988, pp. 341–348.
[15] S. Y. Foo, Y. Takefuji, and H. Szu, "Job-shop scheduling based on modified Tank-Hopfield linear programming networks," *Engineering Application and Artificial Intelligent*, vol. 7, no. 3, pp. 321–327, 1994.
[16] D. N. Zhou, V. Charkassky, T. R. Baldwin, and D. E. Olson, "A neural-network approach to job-shop scheduling," *IEEE Trans. Neural Networks*, vol. 2, no. 1, pp. 175–179, 1991.
[17] C.-S. Zhang and P.-F. Yan, "Neural network method of solving job-shop scheduling problem," *ACTA Automation Sinica*, vol. 21, pp. 706–712, 1995.
[18] T. M. Willems, "Neural networks for job-shop scheduling," *Contr. Eng. Practice*, vol. 2, no. 1, pp. 31–39, 1994.
[19] T. M. Willems and L. E. M. W. Brandts, "Implementing heuristics as an optimization criterion in neural networks for job-shop scheduling," *J. Intell. Manufacturing*, vol. 6, pp. 377–387, 1995.
[20] H.-B. Yu, "Research of Intelligent Production Scheduling Methods and Their Applications," Ph.D. dissertation, Northeastern University of P.R. China, 1997.
[21] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. New York: Wiley, 1982.
[22] H.-L. Fang, P. Ross, and D. Corne, "A promising genetic algorithm approach to job-shop scheduling, rescheduling and open-shop scheduling problems," in *Proc. 5th Int. Conf. Genetic Algorithms*, vol. 67, 1994, pp. 81–100.
[23] S. C. Graves, "A review of production scheduling," *Operations Res.*, vol. 29, pp. 646–675, 1981.
[24] R. H. Nielsen, *Neurocomputing*. Reading, MA: Addison-Wesley, 1989.
[25] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, pp. 4–22, Apr. 1987.
[26] J. F. Muth and G. L. Thompson, *Industrial Scheduling*. Englewood Cliffs, NJ: Prentice-Hall, 1963.

Shengxiang Yang was born in Anhui Province, China, in 1972. He received the B.S. and M.S. degrees in automatic control and the Ph.D. degree in control theory and control engineering from the Northeastern University, Shenyang, China, in 1993, 1996 and 1999, respectively.

He is now a Postdoctoral Research Associate at Department of Computer Science, King's College London, University of London, U.K. His current research interests include artificial neural networks, production scheduling, combinational optimization, genetic algorithms, and network flow theory and algorithms.

Dingwei Wang received the Ph. D. degree in control theory and application from the Northeastern University, Shenyang, China.

He has been a Postdoctoral Fellow at North Carolina State University, Raleigh. He is currently Professor at the Department of Systems Engineering, School of Information Science and Engineering, Northeastern University, Shenyang, China. He has authored three books and has had more than 100 papers published in international and domestic journals, including the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, *International Journal of Production Research*, *Fuzzy Sets and Systems*, etc. His current research interests include MRP-II, JIT manufacturing systems, production planning and scheduling, fuzzy optimization, and genetic algorithms.

Dr. Wang is a member of the New York Academy of Sciences and a member of Automatic Association of China. He serves on the editorial boards of Chinese Journal of Control and Desision.