A Cooperative Coevolutionary Algorithm with Application to Job Shop Scheduling Problem

Zhou Hong, Wang Jian

***Abstract*—An improved cooperative coevolutionary algorithm, which aims at solving job shop scheduling problem, is proposed in this paper. According to the number of machines, population is naturally divided into some subpopulations whose individuals encode the preference list of jobs. The proposed algorithm introduces steady-state reproduction to crossover and mutation operators, and inserts some new individuals to the subpopulation at some other generations, and uses the improved preference-list-based G&T algorithm to decode the whole solutions to calculate fitness by three types of cooperative partners, and adopts an innovative updating technique to speed up the convergence. The optimization results of numerical experiments have shown that, the proposed algorithm has outperformed traditional genetic algorithms and showed strong competition with other heuristics.**

***Index Terms*—Coevolution, Cooperative Partner, Job Shop, Scheduling.**

I. INTRODUCTION

The job shop scheduling problem (JSP) is a simplified model of many actual production scheduling problems and typically a NP-hard problem[1][2]. Because of its inherent intractability, the traditional gradient-based programming optimization algorithms are difficult to solve it. Then some heuristic procedures using priority rules become an attractive alternative, such as priority dispatching heuristics and shifting bottleneck heuristics (SB)[3]. In the recent years, researches have been interested in probabilistic local search methods, for example, simulated annealing (SA)[4], tabu search (TS)[5] and genetic algorithms (GA)[6]-[12] that are the most popular among them.

Genetic algorithms are a parallel random adaptive optimization technology that is based on the survival of the fittest. The population of individuals encoding the solutions evolves and finally converges to the fittest individual representing the optimal solution or the satisfying solution. Because genetic algorithms do not concern the cooperative relationship of isolated individuals, they are prone to premature convergence and slow convergence. However, cooperative coevolutionary algorithms introduce the concept of ecosystem into the traditional evolutionary algorithms, which map the problem into the ecosystem consisting of tow or more interacting species[13]-[16]. To improve the performance of solving JSP, this paper proposes an improved cooperative coevolutionary algorithm, which divides population into some subpopulation according to the number of machines and applies three types of cooperative partners to evaluate the individuals. The optimization results of numerical experiments have showed the proposed algorithm have offset the shortcomings of the tradition genetic algorithms and showed the strong competition with other heuristics.

Manuscript received January 20, 2006. The research is supported by the Natural Science Foundation of China (Grant No. 70371005 and No. 70521001) and New Century Excellent Talents in University of China.

Zhou Hong is with School of Economics and Management, Beihang University, Beijing, P.R.China(corresponding author to provide phone: 086-010-82339458; e-mail: h_zhou@buaa.edu.cn).

Wang Jian is with School of Economics and Management, Beihang University, Beijing, P.R.China(corresponding author to provide phone: 086-010-82339338; e-mail: wangjianpwz@yahoo.com.cn).

II. DESCRIPTION OF JSP

A set of n jobs and m machines are given; every job has a set of operations that have deterministic processing time and have to be processed on a definite machine in a specified sequence. The objective is to determine starting time or finished time of every operation that complies with processing constraints in order to make the given performance index optimal.

There are some constraints for jobs and machines: a job does not access a same machine twice; there are no preference constraints among operations of different jobs; operation can not be interrupted; every machine can process only a job at a time; neither release times or due dates are specified.

The JSP with a makespan objective can be formulated as follows:

$$\min \max_{1\le k\le m}\{\max_{1\le i\le n} c_{ik}\} \tag{1}$$

$$s.t.\quad c_{ik} - p_{ik} + M(1-a_{ihk}) \ge c_{ih} \tag{2}$$

$$c_{jk} - c_{ik} + M(1-x_{ijk}) \ge p_{jk} \tag{3}$$

$$c_{ik} \ge 0 \tag{4}$$

$$x_{ijk} = 0 \text{ or } 1 \tag{5}$$

$$a_{ihk} = 0 \text{ or } 1 \tag{6}$$

* 0-7803-8125-4/03/$17.00 © 2006 IEEE

Where i=1,2 , …,n; h=1,2 , …,m; k=1,2 , …,m; Symbol c_{ik} and p_{ik} denote the completion time and the process time of job i on the machine k; M is an enormous positive number; a_{ihk} and x_{ijk} are indicator coefficient and indicator variable respectively. If processing on machine h preceeds that on machine k for job i, a_{ihk}=1; otherwise a_{ihk}=0. If job i procedes job j on machine k, x_{ijk} =0; otherwise x_{ijk}=0. Eq(1) is the objective function; Eq(2) ensures that the processing sequence of operations for each job corresponds to the prescribed order; Eq(3) ensures that each machine can process only one job at a time.

III. Improved Cooperative Coevolutionary Algorithm

The structure of the cooperative coevolutionary algorithm is followed. The source problem is divided into several subtasks that can be solved by the single genetic algorithm. These subtasks communicate with one another within a shared domain model. A genetically isolated subpopulation represents a subtask and individuals of every subpopulation represent a part of the whole solution. To evaluate individuals from one subpopulation, cooperation is formed with partners from each of the other subpopulations to make up a whole solution that is decoded to calculate the fitness.

A. Dividing and encoding

According to the number of machines the population is naturally divided into m subpopulations and each subpopulation corresponds to one special machine for the n-job m-machine scheduling problem. Individuals of each subpopulation encode the preference list of n jobs and each gene identifies an operation that has to be processed on the corresponding machines. The earlier the job ranks in the preference list, the earlier the job would be processed. However, individuals do not describe the operation sequence on the machine. The actual scheduling is deduced from the whole solution through a simulation which analyzes the state of the waiting queues in front of the machines. This kind of encoding can avoid deadlock and improve the efficiency.

B. Initialization and diversity

The initial solution plays a critical role in determining the quality of the final solution. However, since the initial population has been produced randomly in most heuristics, it not only requires longer search time to obtain the optimal solution but also decrease the search possibility for an optimal solution. In this paper we introduce some simple heuristic knowledge into the initialization. The job whose first schedulable operation can be processed on the corresponding machine is ranked the first gene of individuals form the corresponding subpopulation. If more than one such job exit, one job should be chosen randomly. Thus all jobs can be scheduled as possible as early, which contribute a lot to the objective function. Moreover, concerning only the first gene of individuals not only can not increase the complexity but also can contribute a lot to the quality of the final solution and the early speed of convergence.

In addition, to prevent the proposed algorithm form premature convergence, this paper inserts some new individuals into each subpopulation to replace some old individuals whose fitness values rank behind. Thus the diversity of subpopulations can make the whole solution jump out of the local optimum and continue searching better orientation.

C. Genetic operators

This paper applies steady-state reproduction to crossover operator and mutation operator, which compares the fitness value of children with that of parents, if the fitness value of children is higher than that of parents, replace the parents with the children, or else hold the parents. The introduction of steady-state reproduction can preserve the good individuals of parent generation and prevent the good individuals from the destruction by the genetic operators. We use fitness-based roulette to choose parent individuals, which can fully mine the efficient information of the good individuals with bigger possibility. The fitness values of individuals which belongs to (0, 1) is calculate by Eq(7).

$$f_q(s)=\frac{g_q(s)-\left\{\max_{u\in Pop[q]} g_q(u)+1\right\}}{\min_{u\in Pop[q]} g_q(u)-\left\{\max_{u\in Pop[q]} g_q(u)+1\right\}} \tag{7}$$

Where f_q(s) is the fitness value of the sth individual of subpopulation $Pop[q]$(q=1,2 , …,M); $g_q(u)$ is the objective value of the uth individual of subpopulation $Pop[q]$ by the cooperation with partners. Eq(7) rescales the value of $g_q(s)$ so that it can make the selection effectively and deal with the minimum problem.

In JSP relative positions between genes are important. This paper applies the linear order crossover (LOX) which preserves as much as possible the relative positions between genes and the absolute positions relative to the extremes of the individuals. It applies the swap mutation, which exchange the two genes of different position.

D. Cooperative partners selection

To evaluate one individual from one of subpopulations, we first need to select cooperative partners from every one of the other subpopulations and combined them, along with the individual being evaluated, into a whole solution. Then the whole solution is applied to the objective function. Because the individuals from every subpopulation have a strong inter-activity in the decoding process, this paper applied three selection ways: the best individual from the last evaluation process, a random individual and the individual which stays the same position as the individual being evaluated. We evaluate the current individual with the three different types of cooperative partners and assign the highest fitness value to the current individual's fitness. The method of selecting cooperative partners can not only lower the selection pressure but also increase the number of cooperative partners per subpopulation. Therefore it can improve the performance of the algorithm to some extent.

E. Decoding

This paper applies the preference-list-based G&T algorithm to decode the whole solution that is obtained by selecting the cooperative partners. The improved G&T algorithm does not depend on the priority rules and just use the preference list to select the operation from the conflicting set. It can always generate feasible active schedule. It works as followed:

Step1: let t=0 and begin with PS_t as a null partial schedule. Initially S_t includes all operations with no predecessors.

Step2: determine $\phi_t^* = \min_{i \in S_t}\{\phi_i\}$ and the machine m^* where Φ_t^* can be realized. If more than one such machine exits, the tie is broken by a random choice.

Step3: Construct a conflicting set C_t including all operations which are ready to work on machine m^* and satisfy $\sigma_i<\Phi_t^*$. According to the preference list on the machine m^*, from C_t select the operation which appears first in the preference list and add this operation into PS_t, thus creating a new partial schedule PS_{t+1}.

Step4: For PS_{t+1}, update set S_t as follows:

Step4.1: Delete operation i from set S_t.

Step4.2: Form S_{t+1} by adding the immediate successor of operation i into S_t.

Step4.3: t=t+1.

Step5: Go to Step2 until a complete schedule is generated.

Where PS_t is a partial schedule including t scheduled operations; S_t is a set of schedulable operations of every job at stage t; σ_i is the earliest time when operation $i \in S_t$ could be started; Φ_i is the earliest time when operation $i \in S_t$ could be completed; C_t is the set of conflicting operations. For the given partial active schedule, the potential starting time σ_i is the bigger value of the completing time of the direct predecessor of the operation i and the latest completing time on the machine required by operation i; The potential finishing time is simply $\Phi_i=\sigma_i+t_i$, where t_i is the processing time of operation i.

Let's take an example of a three-job three-machine scheduling problem presented in Table 1 to illustrate the decoding process.

TABLE 1
EXAMPLE OF A THREE-JOB THREE-MACHINE SCHEDULING PROBLEM

Job	Processing Time: Operation 1	2	3	Job	Machine Sequence: Sequence 1	2	3
j_1	3	3	2	j_1	m_1	m_2	m_3
j_2	1	5	3	j_2	m_1	m_3	m_2
j_3	3	2	3	j_3	m_2	m_1	m_3

Suppose to evaluate the individual (1, 3, 2) from subpopulation 1. The cooperative partners (2, 3, 1) and (2, 1, 3) are selected from subpopulation 1 and subpopulation 3, along with the being evaluated individual, to form the whole solution below.

$$\begin{matrix} m_1 \\ m_2 \\ m_3 \end{matrix}\begin{bmatrix} 2 & 3 & 1 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$

The rows of the matrix identify the machines and the columns for the operations. The preference list on a special machine does not describe the operation sequence. The actual scheduling is deduced from the whole solution through a simulation which analyzes the state of the waiting queues in front of the machines and, if necessary, uses the preference list to choose the operation which appears first.

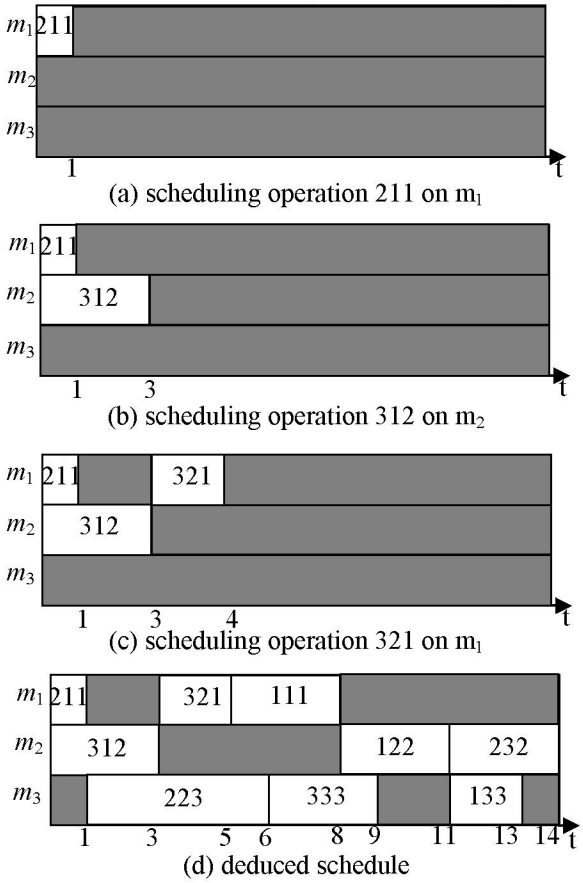

Figure 1 Deduce a Schedule Through the Preference-list-based G&T Algorithm

At the initialization phase, data sets are as followed:

$$PS_0 = \phi,\ S_0 = \{o_{111}, o_{211}, o_{312}\}$$
$$\phi_1^* = \min\{3,1,3\} = 1,\ m^* = 1$$
$$C_1 = \{o_{111}, o_{211}\}$$

Now operation o_{111} and operation o_{211} compete for machine m_1. Because job j_2 is ranked before job j_1 on the machine 1 in the above whole solution, operation o_{211} is scheduled on machine m_1 as showed in Figure 1(a). After updated, data sets are as followed:

$$PS_1 = \{o_{211}\},\ S_1 = \{o_{111}, o_{223}, o_{312}\}$$
$$\phi_1^* = \min\{4,6,3\} = 3,\ m^* = 2$$
$$C_1 = \{o_{312}\}$$

Only operation o_{312} asks for machine m_2, so operation o_{312} is scheduled on machine m_2 as showed in Figure 1(b). After updated, data sets are as followed:

$$PS_2 = \{o_{211}, o_{312}\},\ S_2 = \{o_{111}, o_{223}, o_{321}\}$$
$$\phi_2^* = \min\{4,6,5\} = 3,\ m^* = 1$$

$$C_2 = \{o_{111}, o_{321}\}$$

Now operation o_{111} and operation o_{321} compete for machine 1. Because job j_3 is ranked before job j_1 on machine m_1 in the above whole solution, operation o_{321} is scheduled on machine m_1 as showed in Figure 1(c). Iterate these steps until a complete schedule is deduced from the given whole solution as showed in Figure 1(d). The final operation sequence of all jobs on all machines is as followed:

$$\begin{matrix} m_1 \\ m_2 \\ m_3 \end{matrix}\begin{bmatrix} 2 & 3 & 1 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}$$

F. Updating technology

Individuals of every subpopulation do not describe the actual schedule on the relevant machine because they are preference lists, so the final operation sequence that is deduced through the decoding process is possibly different from the preference list. In section 3.4 the preference list on machine m_2 is (1, 3, 2), but the final operation sequence on machine m_2 is (3, 1, 2).

Let C be the space of the whole solutions and let S be the space of active schedules and let F: $C \rightarrow S$ be the function which generates an active schedule starting from a whole solution. $a=\{a_{ij}\} \in C$ is a matrix whose rows identify the machines and columns for the operations. If $F(a)=b$, then $F(b)=b$. A schedule b is deduced from a whole solution a, then if the schedule b can be read itself as a whole solution b and be decoded, we can obtain the schedule b itself again. From the decoding process we can see that the different whole solutions can generate the same final operation sequence. That is to say, a small space C^* exits and any whole solution $a \in C^* \in C$ can generate the same schedule b and $b \in C^* \in C$.

It can be concluded that only the whole solution b in C^*, which is the same as the active schedule b, has as much as possible relationship among all these individuals and effective gene information that would be transmitted to the children. This paper substitutes the individual being evaluated from a given subpopulation with the final operation sequence on the corresponding machine. For example, after a given individual (1, 3, 2) from subpopulation 2 is evaluated in section 3.4, we update the individual by substituting it with the final operation sequence (3, 1, 2) on machine m_2. The optimization results have showed this updating technology is a critical procedure for the proposed algorithm, which can greatly improve the efficiency.

G. Algorithm framework

The improved cooperative coevolutionary algorithm showed in Figure 2 works as followed:

Step1: Generate M sets of initial subpopulations $Pop[m]$ (m=1,2 , …,M).

Step2: Initially evaluate each individual of each subpopulation and set f_{best} to be the best fitness value of all individuals. The random individuals and the individuals that stay the same position as the individual being evaluated are selected to be cooperative partners.

Step3: Use the updating technology to substitute the individual being evaluated with the final operation sequence on the corresponding machine.

Step4: Coevolve. Set m=1.

Step4.1: Evaluate the sth (s=1, 2 , …,S) individual of subpopulation $Pop[m]$. Set s=1.

Step4.2: Choose the other parent by the fitness-based roulette and apply LOX operator to generate one child.

Step4.3: Apply swap mutation to generate one child.

Step4.4: Calculate the fitness value of the children. The best individuals and the random individuals and the individuals that stay the same position as the individual being evaluated are selected to be cooperative partners.

Step4.5: If the fitness value of child is higher than that of parent, then apply the updating technology. Otherwise, preserve the parent. If the fitness value of the best child is higher f_{best}, then f_{best} is updated with the new best fitness value.

Step4.6: Set $s \leftarrow s+1$. If $s<S$ (the number of individuals from each subpopulation), then go to Setp4.2. Otherwise, go to Step4.7.

Step4.7: Set $m \leftarrow m+1$. If $m<M$ (the number of subpopulations), then go to Step4.1. Otherwise, go to Step5.

Step5: If the termination criteria are satisfied, then stop. Otherwise, go to Step4.

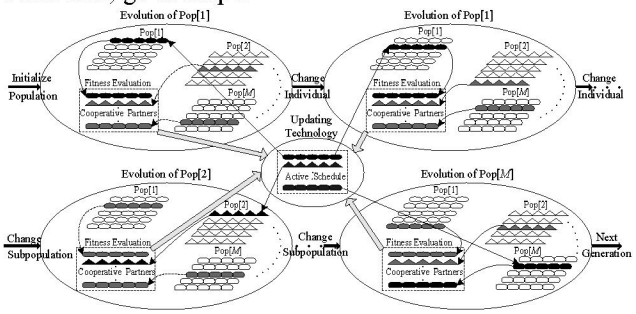

Figure 2. Improved Cooperative Cooevolutionary Algorithm

IV. Experimental Results and Analysis

The improved cooperative coevolutionary algorithm (ICCA) was employed to solve some series of job shop benchmark problems. In our experiment, the number of subpopulations is determined by the number of machines, subpopulation size is 30, crossover rate and mutation rate are 1, generation number is 400, 25 new individual are inserted into each subpopulation every other 5 generations. Each benchmark problem was run 10 times by ICCA on a PC with P4 3.0 and 512 MB main memory.

The optimization result of ICCA was compared with that of several variants of genetic algorithms on MT problem in Table 2. ICCA did not seek the optimal value on the MT10 and MT20 problems, but it sought 935 and 1173 respectively, which outperformed the best value that several variants of genetic algorithms sought.

TABLE 2

COMPARISON WITH SEVERAL VARIANTS OF GAS ON MT PROBLEM

Authors	Problem		
	MT06	MT10	MT20
Optimum	55	930	1165
Nakano and Yamada	55	965	1215
Gen	55	962	1175
Fang	-	949	1189
Dorndorf and Pesch	55	938	1178
Croce	55	946	1178
Cheng	55	948	1196
Bierwirth	55	936	1181
ICCA	55	935	1173

Figure 3 showed the evolution process of ICCA and standard genetic algorithms (SGA) on the MT10. Both algorithms applied the evolution strategy this paper described, but used the optimal parameters respectively. ICCA sought 957 just at the twentieth generation and continued seeking new better value as the time proceeded, but SGA sought 958 at sixtieth generation and tracked in local optimum. Therefore ICCA improved the early speed of convergence and the optimum-seeking ability for SGA.

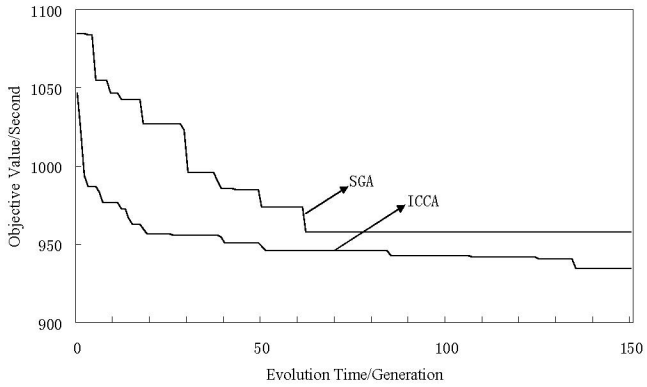

Figure 3. Evolution process of ICCA and SGA on MT10 problem

TABLE 3
COMPARISON WITH OTHER HEURISTICS ON LA PROBLEM

Problem	Method				
	Optimum	SA	TS	SB	ICCA
LA01	666	666	666	666	666
LA06	926	926	926	926	926
LA11	1222	1222	1222	1222	1222
LA16	945	956	945	978	945
LA21	1046	1063	1048	1084	1064
LA26	1218	1218	1218	1224	1226
LA31	1784	1784	1784	1784	1784
LA36	1268	1293	1278	1305	1292

The optimal result of ICCA was compared with that of simulated annealing (SA) and tabu search (TB) and shifting bottleneck (SB) on LA problem in Table 3. ICCA sought the optimal value on LA01, LA06, LA11 and LA31 problems. Moreover, ICCA sought good values compared with the several heuristics on LA21, LA26 and LA36 problem. It can be seen that ICCA showed strong competition with other heuristics.

V. CONCLUSIONS

An improved cooperative coevolutionary algorithm, which aims at solving job shop scheduling problem, is proposed in this paper. According to the number of machines, population is naturally divided into some subpopulations whose individuals encode the preference list of jobs waiting for the corresponding machine. When each subpopulation is initialized, the job whose first schedulable operation could be processed on the corresponding machine will be ranked the first gene of individuals. The proposed algorithm applies steady-state reproduction to crossover and mutation operators, and inserts some new individuals to the subpopulations at some other generations, and uses the improved preference-list-based G&T algorithm to decode the whole solutions to calculate fitness by three types of cooperative partner selection, and adopts an innovative updating technique to speed up the convergence. Numerical experiments have been made which employ the improved algorithm to solve some job shop benchmark problems. The optimization results have shown that, our proposed algorithm has outperformed traditional genetic algorithms and showed strong competition with other meta-heuristic algorithms.

How to decompose complex problem and sample the cooperative partners to evaluate the individual efficiently would be critical questions to be solved on the scheduling problem.

REFERENCES

[1] Wang Ling, *Shop scheduling with genetic algorithms*, Beijing: Tsinghua university publisher, 2003.

[2] M. Gen and R. Cheng, *Genetic algorithms and engineering optimization*, New York: John Wiley & Sons, 2000.

[3] J. Adams, E. Balas and D. Zawack. "The shifting bottleneck procedure for job shop scheduling", *Management Science*, Vol. 34, pp. 391-401, 1988

[4] P. J. M. Van Laarhoven, E. H. L. Aarts and J. K. Lenstra, "Job shop scheduling by simulated annealing", *Operations Research*, Vol.40, pp.113-125, 1992.

[5] M. Dell'Amico and M. Trubian, "Applying tabu search to the job shop scheduling problem", *Annals of Operations Research*, Vol. 41, pp.231-252, 1993.

[6] R. Nakano and T. Yamada, "Conventional genetic algorithms for job shop problem", *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 474-479, 1991.

[7] M. Gen and R. Cheng, *Genetic algorithms and engineering design*, New York: John Wiley & Sons, 1997.

[8] H. Fang, P. Ross and D. Corne, "A promising genetic algorithm approach to job-shop scheduling, rescheduling and open-shop scheduling problem", *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 375-382, 1993.

[9] U. Dorndorf and E. Pesch, "Evolution based learning in a job shop scheduling environment", *Computers and Operations Research*, Vol. 22, pp. 25-40, 1995.

[10] F. D. Croce, R. Tadei and G. Volta, "A genetic algorithm for the job shop problem", *Computers and Operations Research*, Vol. 22, pp. 15-24, 1995.

[11] R. Cheng, "A study on genetic algorithm-based optimal scheduling techniques", Ph.D. thesis, Tokyo Institute of Technology, 1997.

[12] C. Bierwirth, "A generalized permutation approach to job shop scheduling with genetic algorithm", *OR-Spektrum*, Vol. 17, pp. 87-92, 1995.

[13] M. Potter and K. De Jong, "Cooperative coevolution: an architecture for evolving coadapted subcomponents", *Evolutionary Computation*, Vol. 8, pp. 1-29, 2000.

[14] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization", *Proceedings of the Third Parallel Problem Solving from Nature*, pp.249-257, 1994.

[15] R. P. Wiegand, W. C. Liles and K. De Jong, "An empirical analysis of collaboration methods in cooperative algorithms", *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1235-1242, 2001.

[16] Y. K. Kim, K. Park and J. Ko. "A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling", *Computers and Operations Research*, Vol. 30, pp. 1151-1171, 2003.