World Academy of Science, Engineering and Technology
International Journal of Economics and Management Engineering
Vol:5, No:11, 2011

Hybrid Artificial Immune System for Job Shop Scheduling Problem

Bin Cai, Shilong Wang, and Haibo Hu

***Abstract*—**The job shop scheduling problem (JSSP) is a notoriously difficult problem in combinatorial optimization. This paper presents a hybrid artificial immune system for the JSSP with the objective of minimizing makespan. The proposed approach combines the artificial immune system, which has a powerful global exploration capability, with the local search method, which can exploit the optimal antibody. The antibody coding scheme is based on the operation based representation. The decoding procedure limits the search space to the set of full active schedules. In each generation, a local search heuristic based on the neighborhood structure proposed by Nowicki and Smutnicki is applied to improve the solutions. The approach is tested on 43 benchmark problems taken from the literature and compared with other approaches. The computation results validate the effectiveness of the proposed algorithm.

***Keywords*—**Artificial immune system, Job shop scheduling problem, Local search, Metaheuristic algorithm

I. Introduction

THE job shop scheduling problem (JSSP) is one of the most difficult problems in combinatorial optimization that has garnered considerable attention due to both its practical importance and its solution complexity. Efficient methods for solving the JSSP have significant effects on profitability and product quality. During the last three decades, many solution methods have been proposed to solve the JSSP. Those approaches can be divided into two categories: exact methods and approximation algorithms. Exact methods, such as branch and bound, linear programming and decomposition methods, guarantee global convergence and have been successful in solving small instances. In manufacturing systems, most scheduling problems are very complex in nature and very complicated to be solved by exact methods to obtain a global optimal schedule. For the big instances there is a need for approximation algorithms, which include priority dispatch, shifting bottleneck approach, local search, and heuristic methods. Recently, using a high-level strategy to guide other heuristics, known as metaheuristics, led to better and more appreciated results in a relatively short period. Therefore, a number of metaheuristics were proposed in literature for the past two decades to deal with the JSSP such as genetic algorithm (GA) [1], simulated annealing (SA) [2], taboo search (TS) [3], artificial immune system (AIS) [4]-[6], greedy randomized adaptive search procedure (GRASP) [7] etc. A comprehensive survey of job shop scheduling techniques has been done by Jain and Meeran [8].

Artificial Immune System is an evolutionary computation technique inspired by the biological immune system. Several concepts from the immune system have been extracted and applied for solution to real world science and engineering problems. In recent years there have been a lot of reported works focusing on the AIS in [4]-[6], [9]-[11], which has been applied widely in the function optimization and some other fields. As AIS became popular in the mid 1990s, many researchers started to apply this metaheuristic method to the JSSP. Hart[4] tackled a job shop scheduling problem by an AIS approach in which an antibody represents a scheduling, while an antigen represents a set of changes that can occur and cause the schedule to be modified. Coello[6] proposed a clonal selection based algorithm to solve JSSP. Due to the NP-hard nature of the JSSP, using simple AIS to solve the difficult problem may not be efficient in practice. Much effort in the literature has focused on hybrid methods. Tsai, Ho, Liu and Chou[12] proposed an improved immune algorithm for job shop scheduling problem. Ge, Sun, Liang and Qian [13] designed a hybrid algorithm of combining particle swarm optimization and AIS for solving JSSP.

In this paper, an effective hybrid intelligent algorithm for JSSP based on artificial immune system and local search is presented. The remainder of the paper is organized as follows. An introduction for the job shop scheduling problem is given in Section II. Detailed description of the proposed job shop scheduling algorithm is presented in Section III. Section IV discusses the experimental results. Finally, we summarize the paper and present our future work in Section V.

II. Job Shop Scheduling Problem

The problem studied in the paper is a deterministic and static *n*-job, *m*-machine JSSP. In this problem, *n* jobs are to be processed by *m* machines. Each job consists of a predetermined sequence of task operations, each of which needs to be processed without preemption for a given period of time on a given machine. Tasks of the same job cannot be processed concurrently and each job must visit each machine exactly once. Each operation cannot be commenced until the processing is completed, if the precedent operation is still being processed. A schedule is an assignment of operations to time slots on the machines. The makespan is the maximum completion time of the jobs. The objective of the JSSP is to find

Bin Cai is with the School of Software Engineering and is a PhD candidate at the State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing, 400030, China (*Corresponding author, phone:+862365127222; fax:+862365678333; e-mail: caibin@cqu.edu.cn).

Shilong Wang is with the State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing, 400030, China. (e-mail: slwang@cqu. edu.cn).

Haibo Hu is with the School of Software Engineering, Chongqing University, Chongqing, 400030, China (e-mail: hbhu@cqu. edu.cn).

a schedule that minimizes the makespan.

Explaining the problem more specifically, let $J=\{1, 2,..., n\}$ denote the set of jobs, $M=\{1,2,\ldots,m\}$ represent the set of machines, and $O=\{0, 1, 2, \ldots, n\times m, n\times m+1\}$ be the set of operations to be scheduled, where 0 and $n\times m+1$ represent the dummy initial and final operations, respectively. The operations are interrelated by the precedence constraints, which force each operation j to be scheduled after all predecessor operations E_j are completed. Moreover, operation j can only be scheduled if the required machine is idle. Furthermore, let p_j and c_j denote the fixed processing time and the finish time of operation j, respectively. Let $B(t)$ be the set of operations being processed at time t, and let $\theta_{jm}=1$ if operation j is required to process on machine m ($\theta_{jm}=0$ otherwise).

The conceptual model of the JSSP can be stated as [14]

$$\min c_{n\times m+1} \tag{1}$$

$$s.t.\ \ c_k \le c_j - p_j,\ j=1,2,\ldots,n\times m+1,\ k\in E_j \tag{2}$$

$$\sum_{j\in B(t)} \theta_{jm} \le 1,\ m\in M,\ t\ge 0 \tag{3}$$

$$c_j \ge 0,\ j=1,2,\ldots,n\times m+1 \tag{4}$$

The objective function (1) minimizes the finish time of the last operation, namely, the makespan. Constraint (2) imposes the precedence relations between operations. Constraint (3) represents that one machine can only process one operation at a time, and constraint (4) forces the finish times to be nonnegative.

III. Hybrid Artificial Immune System

The artificial immune system is based on two main principles [15]: clonal selection and affinity maturation principles. In the first principle, antibodies that have better affinities are selected for reproduction and the numbers of clones of each antibody is proportional to its affinity value. The latter principle consists of two main processes: hypermutation and receptor editing. The hypermutation operator performs an affinity maturation process inversely proportional to the fitness values generating the matured clone population. After cloning, sorting and deleting the repetition, the receptor editing process is conducted by eliminating antibodies from the population based on the desired percentage of antibody elimination. The whole process is repeated until the termination criterion is satisfied. Before AIS can be run, a suitable representation for the problem must be devised. A fitness function is also required, which assigns a figure of merit to each encoded solution. During the run, antibodies must be selected for cloning and hypermutation to generate offspring.

A. Antibody Representation

In solving JSSP using AIS, the first thing is the representation of the problem. Each antibody represents a possible solution to the problem. Some popular representations for solving JSSP are: operation based, job based, preference list based, priority rule based, and job pair relationship based representations [16]. In this paper, an operation based representation is adopted, which uses an unpartitioned permutation with m-repetitions of job numbers for problems with n jobs and m machines. Within the representation, each job number occurs m times in the antibody. By scanning the antibody from left to right, the k-th occurrence of a job number refers to the k-th operation in the technological sequence of this job.

For example, suppose that a antibody is given as [2 3 1 1 3 3 1 2 2] in a three jobs and three machines problem. Because each job consists of three operations, the job number occurs exactly three times in the antibody. The fifth gene of the permutation implies the second operation of job 3 because number 3 has been repeated twice. Similarly, the sixth gene represents the third operation of job 3, and so on. The prominent advantage of operation based representation is that the permutation is always feasible. Moreover, it eliminates the deadlock schedules that are incompatible with the technological constraints and can never be finished. However, it will produce redundancy in the search space and will cause the search-space size to expand to $(n\times m)!/(m!)^n$.

B. Antibody Decoding

In general, schedules can be classified into three types: semiactive schedule, active schedule and non-delay schedule [17]. Semiactive schedules contain no excess idle time, but they can be improved by shifting some operations to the front without delaying others. Active schedules contain no idle time, and no operation can be finished earlier without delaying other operations. The set of non-delay schedules is a subset of active schedules. In a non-delay schedule, no machine is kept idle at a time when it could begin processing other operations. In order to further reduce the solution space, Zhang, Rao and Li [18] proposed a new type of schedule: full active schedule (FAS), which can be defined as a schedule with no more permissible left shifts and right shifts. Fig. 1 shows the relationships between the classes of schedules. The optimal schedule is guaranteed to be a full active schedule. Therefore, we only need to find the optimum solution in the set of full active schedules.

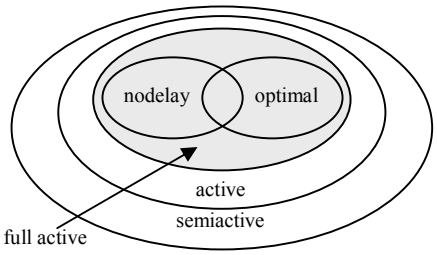

Fig. 1 Classes of schedules

The objective of the antibody decoding procedure is to transform the antibodies to schedules and obtain their makespans. An active schedule is decoded from a antibody with the following decoding procedure: firstly translate the antibody to a list of ordered operations, and then generate the schedule by a one-pass heuristic based on the list. The first operation in the list is scheduled first, then the second operation,

and so on. Each operation under treatment is allocated in the best available processing time for the corresponding machine the operation requires. The process is repeated until all operations are scheduled. Using the same algorithm to the active schedule, we can get a full active schedule with only small modifications. By reversing the antibody based on the operation based representation and all of the technological sequences, a given schedule can be converted to another schedule. The new schedule is equivalent to the original one with the same makespan (the same critical path) and the reversed antibody (i.e., reversed job processing sequences on same machine). Through left shifting the new schedule, we can obtain the makespan and antibody of the full active schedule.

C. Affinity Evaluation

Each antibody a has a makespan value $makespan(a)$. Affinity value $f(a)$ of the antibody is calculated from the affinity function. The affinity function is defined as (5).

$$f(a) = 1/makespan(a) \tag{5}$$

From this relation, a lower makespan value gives a higher affinity value. Further the cloning of antibodies is done directly proportional to their affinity function values. Therefore, there will be more clones of antibodies that have lower makespan values than those with higher makespan values in the new generated clone population.

D. Proliferation and hypermutation

Proliferation operation is to generate copies of every individual in an antibody population proportionally to its affinity value. The amount of clones of antibody a_i is calculated according to (6).

$$n_i = round(N_c \cdot f(a_i) / \sum_{i=1}^{pop\_size} f(a_i)) \tag{6}$$

Where N_c is a given value relating to the clone scale. $f(a_i)$ is the affinity of the antibody a_i, pop_size is the size of population. $round()$ is the operator that rounds its argument towards the closest integer. Obviously, the higher the affinity is, the greater the number of copies is, and vice versa.

After producing clones, the hypermutation stage is implemented. Two types of mutation operators named insertion mutation and displacement mutation are used for generated clones. In this work, the two mutation operators alternate randomly with equal probability. Two mutations are described as follows:

1) Insertion mutation selects two elements randomly and inserts the back one before the front one.

2) Displacement mutation selects a substring randomly and inserts it in a random position.

E. Selection and Receptor editing

After cloning and mutation processes, a number of the antibodies in the antibody population are eliminated and randomly created antibodies replace with them. This mechanism allows finding new schedules that correspond to new search regions in the total search space. In this paper, antibodies with different affinity values are selected as a candidate solution population. If the size of the candidate population is less than the size of population *pop_size*, copy antibodies from the candidate population into new population and fill up the remaining slots of new population with randomly generated antibodies. Otherwise, select the best *pop_size* antibodies from candidate population to new population.

F. Local Search Procedure

Local search techniques have been proven useful in solving combinatorial problems. Local search methods are applied to a neighborhood of a current solution. In the case of JSSP, a neighborhood is achieved by moving and inserting an operation in a machine sequence. In this paper, we focus particularly on the approach of Nowicki and Smutnicki [3], which is noted for proposing and implementing the most restrictive neighborhood in the literature. According to Nowicki and Smutnicki's work, a critical path in the solution is identified first. Then the operations on the critical path are called critical operations and the maximal sequence of adjacent critical operations that are processed on the same machine can be defined as blocks. The neighborhood is defined as interchanges of the last two or the first two critical operations of the blocks if the blocks are neither the first block nor the last block. In the first block only the last two operations and symmetrically in the last black of the critical path only the first two operations are swapped. If a block contains only one operation no swap is made. The Nowicki and Smutnicki's neighborhood is illustrated in Fig. 4.

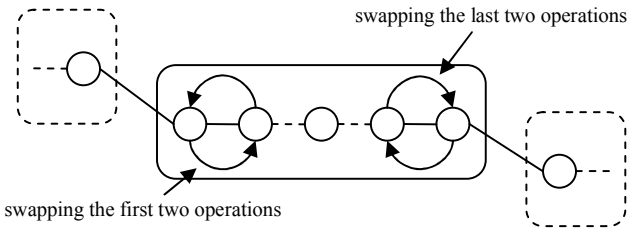

Fig. 4 The Nowicki and Smutnicki's neighborhood

The proposed local search starts with a feasible schedule S as an input. The input schedule is set to S_{best} which stands for the best found solution. Then, a single arbitrary critical path is generated and a neighborhood of schedule S_{best} is constructed. Randomly select a schedule S_{new} from the neighborhood. If S_{new} is better (i.e. has a lower makespan) than S_{best}, the S_{best} is replaced by S_{new}. The procedure is repeated until a maximum number of iterations (*LOC_ITER*) without improving the best found solution is reached.

TABLE I
COMPUTATIONAL RESULTS OF FT AND LA TEST INSTANCES

Size	BKS	HAIS	HIA	AIS	HGA	GRASP	TSAB	Beam Search	RCS	SBII
6×6	55	**55**	55	55	55	55	55	-	55	55
10×10	930	**930**	930	936	930	938	930	1016	930	930
20×5	1165	**1165**	1165	1165	1165	1169	1165	-	1165	1178
10×5	666	**666**	666	666	666	666	666	666	666	666
10×5	655	**655**	655	655	655	655	655	704	655	669
10×5	597	**597**	597	597	597	604	597	650	597	605
10×5	590	**590**	590	590	590	590	590	620	590	593
10×5	593	**593**	593	593	593	593	593	593	593	593
15×5	926	**926**	926	926	926	926	926	926	926	926
15×5	890	**890**	890	890	890	890	890	890	890	890
15×5	863	**863**	863	863	863	863	863	863	863	863
15×5	951	**951**	951	951	951	951	951	951	951	951
15×5	958	**958**	958	958	958	958	958	958	958	959
20×5	1222	**1222**	1222	1222	1222	1222	1222	1222	1222	1222
20×5	1039	**1039**	1039	1039	1039	1039	1039	1039	1039	1039
20×5	1150	**1150**	1150	1150	1150	1150	1150	1150	1150	1150
20×5	1292	**1292**	1292	1292	1292	1292	1292	1292	1292	1292
20×5	1207	**1207**	1207	1207	1207	1207	1207	1207	1207	1207
10×10	945	**945**	945	945	945	946	945	988	945	978
10×10	784	**784**	784	784	784	784	784	827	784	787
10×10	848	**848**	848	848	848	848	848	881	848	859
10×10	842	**842**	842	842	842	842	842	882	848	860
10×10	902	**902**	902	907	907	907	902	948	907	914
15×10	1046	**1048**	1046	1046	1046	1091	1047	1154	1069	1084
15×10	927	**927**	932	927	935	960	927	985	937	944
15×10	1032	**1032**	1032	1032	1032	1032	1032	1051	1032	1032
15×10	935	**938**	950	935	953	978	939	992	942	976
15×10	977	**983**	979	979	986	1028	977	1073	981	1017
20×10	1218	**1218**	1218	1218	1218	1271	1218	1269	1218	1224
20×10	1235	**1247**	1256	1240	1256	1320	1236	1316	1285	1291
20×10	1216	**1216**	1227	1216	1232	1293	1216	1373	1216	1250
20×10	1152	**1174**	1184	1170	1196	1293	1160	1252	1208	1239
20×10	1355	**1355**	1355	1355	1355	1368	1355	1435	1355	1355
30×10	1784	**1784**	1784	1784	1784	1784	1784	1784	1784	1784
30×10	1850	**1850**	1850	1850	1850	1850	1850	1850	1850	1850
30×10	1719	**1719**	1719	1719	1719	1719	1719	1719	1719	1719
30×10	1721	**1721**	1721	1721	1721	1753	1721	1780	1721	1721
30×10	1888	**1888**	1888	1888	1888	1888	1888	1888	1888	1888
15×15	1268	**1275**	1281	1281	1279	1334	1268	1401	1292	1305
15×15	1397	**1397**	1415	1408	1408	1457	1407	1503	1411	1423
15×15	1196	**1202**	1213	1204	1219	1267	1196	1297	1278	1255
15×15	1233	**1233**	1246	1249	1246	1290	1233	1369	1233	1273
15×15	1222	**1224**	1240	1228	1241	1259	1229	1347	1247	1269
Average gap(%)		**0.12**	0.33	0.18	0.40	1.78	0.06	4.35	0.61	1.39
No. of instance		**43**	43	28	43	43	43	41	43	43
o. of BKS obtained		**35**	32	33	31	23	37	18	31	20

The brief outline of the local search algorithm can be described as follows.

Step 1) Set the best found solution $S_{best} = S$. Calculate the makespan $C_{max}(S_{best})$ of S_{best}. Set iteration counter *count* to 1.

Step 2) Repeat Step 3) – 6) until *count* > *LOC_ITER*.

Step 3) Randomly selecte a schedule S_{new} from the neighborhood of S_{best}. Calculate the makespan $C_{max}(S_{new})$ of S_{new}.

Step 4) If $C_{max}(S_{new}) < C_{max}(S_{best})$ go to Step 5), else go to Step 6)

Step 5) Update S_{best} by setting $S_{best} = S_{new}$. Set *count* to 1.

Step 6) Set *count* = *count* + 1.

G.Designing a hybrid artificial immune system for JSSP

The brief outline of the proposed algorithm can be described as follows.

Step 1) Set values of population size *pop_size*, maximum number of iterations *MAX_GEN*, N_c and *LOC_ITER*.

Step 2) Generate a population P_0 with *pop_size* antibodies randomly and evaluate the antibodies with the decoding procedure; set generation counter $g = 1$ and the current population $P_{old} = P_0$.

Step 3) Repeat Step 4) – 9) until $g > MAX\_GEN$.

Step 4) Each antibody in P_{old} will be cloned independently and proportionally to its affinity, generating a repertoire P_c.

Step 5) The repertoire P_c is submitted to an hypermutation process, generating a population P_m.

Step 6) Select antibodies with different affinity from population $P = P_{old} \cup P_m$, generating population P_s.

Step 7) If the size of P_s is less than *pop_size*, copy antibodies from P_s into new population P_{new} and fill up the remaining slots of P_{new} with randomly generate antibodies, otherwise copy the best *pop_size* antibodies from P_s into P_{new}.

Step 8) Implement local search on every antibody in P_{new}.

Step 9) Set $P_{old} = P_{new}$

IV. Computational Results

In this paper, we use 43 instances that are taken from the ORLibrary [19] as test benchmarks to test our new proposed hybrid AIS, named HAIS. In the 43 instances, FT06, FT10 and FT20 were designed by Fisher and Thompson and instances LA01–LA40 that were designed by Lawerence. The algorithm was implemented in C++ and the tests were run on a computer with Pentium IV2.4G and 1GB RAM. In our experiments, population size *pop_size* = 100, $N_c = 3$, *LOC_ITER* is the smallest integer number not less than $n/2$. The algorithm was terminated when after $MAX\_GEN = 2 \times n \times m$ generations of the algorithm, and each instance is randomly run 20 times. Numerical results are compared with those reported in some existing literature works using some heuristic and metaheuristic algorithms, including HIA [13], HGA [14], AIS [6], GRASP [7], TSAB [3], Beam Search [20], RCS [21], and SBII [22].

Table I summarizes the results of the experiments. The contents of the table include the name of each test problem (Instance), the scale of the problem (Size), the value of the best known solution for each problem (BKS), the value of the best solution found by using the proposed algorithm (HAIS) and the best results reported in other research works.

TABLE II
SUMMARY OF RESULTS FOR TYPICAL INSTANCES

Instance	Size	BKS	Best	BRD (%)	Mean	MRD (%)	t-avg (s)
ft06	6×6	55	55	0.00	55	0.00	1.45
ft10	10×10	930	930	0.00	936.55	0.70	16.50
ft20	20×5	1165	1165	0.00	1180.75	1.35	20.36
la01	10×5	666	666	0.00	666	0.00	3.50
la06	15×5	926	926	0.00	926	0.00	9.04
la11	20×5	1222	1222	0.00	1222	0.00	19.96
la16	10×10	945	945	0.00	945.7	0.07	14.71
la21	15×10	1046	1048	0.19	1060.35	1.37	40.83
la26	20×10	1218	1218	0.00	1219.1	0.09	87.65
la31	30×10	1784	1784	0.00	1784	0.00	280.69
la36	15×15	1268	1275	0.55	1286.9	1.49	90.19

It can be seen from Table I that the proposed algorithm is able to find the best known solution for 35 instances, i.e. in about 81% of the instances, and the deviation of the minimum found makespan from the best known solution is only on average 0.12%. The proposed algorithm yields a significant improvement in solution quality with respect to almost all other algorithms, expected for the approach proposed by Nowicki and Smutnicki. The superior results indicate the successful incorporation of the improved AIS and LS, which facilitates the escape from local minimum points and increases the possibility of finding a better solution. Therefore, it can be concluded that the proposed hybrid AIS solves the JSSP fairly efficiently.

As mentioned above, the algorithm is performed 20 times for each instance. Table II lists the best solution (Best), the relative deviation of the best solution (BRD), the mean solutions (Mean), the relative deviation of the mean solution (MRD), and the average computing time (t-avg) of some typical instances with different size. The MRD is commonly zero for small-size problem and is not more than 1.5% for most other problems.

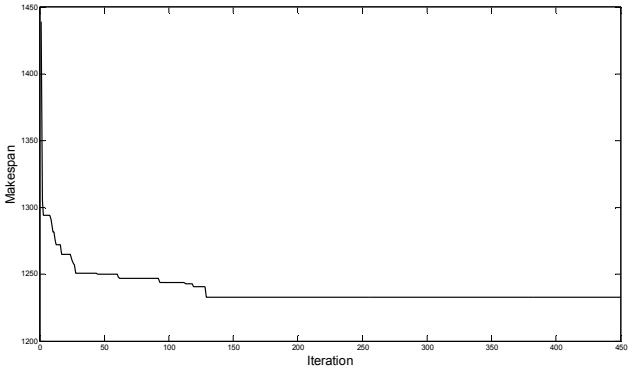

Fig. 5 Representative convergence curve for la39

To illustrate the simulated results more intuitively, the problem la39 as one of the hardest problems is specially described as an example. Fig. 5 plots the representative convergence curve finding best solution. Fig. 6 shows a Gantt chart of a best solution.

V. Conclusion And Perspectives

This paper presents a hybrid algorithm combining artificial immune system with local search for the JSSP. In the algorithm a new selection strategy and receptor editing of artificial immune system for JSSP is designed and a Nowicki and Smutnicki's neighborhood based local search algorithm is incorporated. This allows the AIS to explore more solution space whereas LS does the exploitation part. The approach is tested on a set of 43 benchmark problems taken from the literature and compared with other approaches. The computational results show that the proposed approach produced optimal or near-optimal solutions on all instances tested. Overall, the algorithm produced solutions with an average relative deviation of 0.12% to the best known solution. In our future work we aim to extend the proposed algorithm in order that it can be applied to more practical and integrated manufacturing problems such as dynamic arrivals, machine breakdown, or other factors that affect job status over time.

Acknowledgment

This work is supported by the National Natural Science Funds for Distinguished Young Scholars (No. 50925518), the National S&T Major Project of China under Grant No. 2009ZX07315-006, and the Key NSF of Chongqing City in China under Grant No. CSTC2011BA2022.

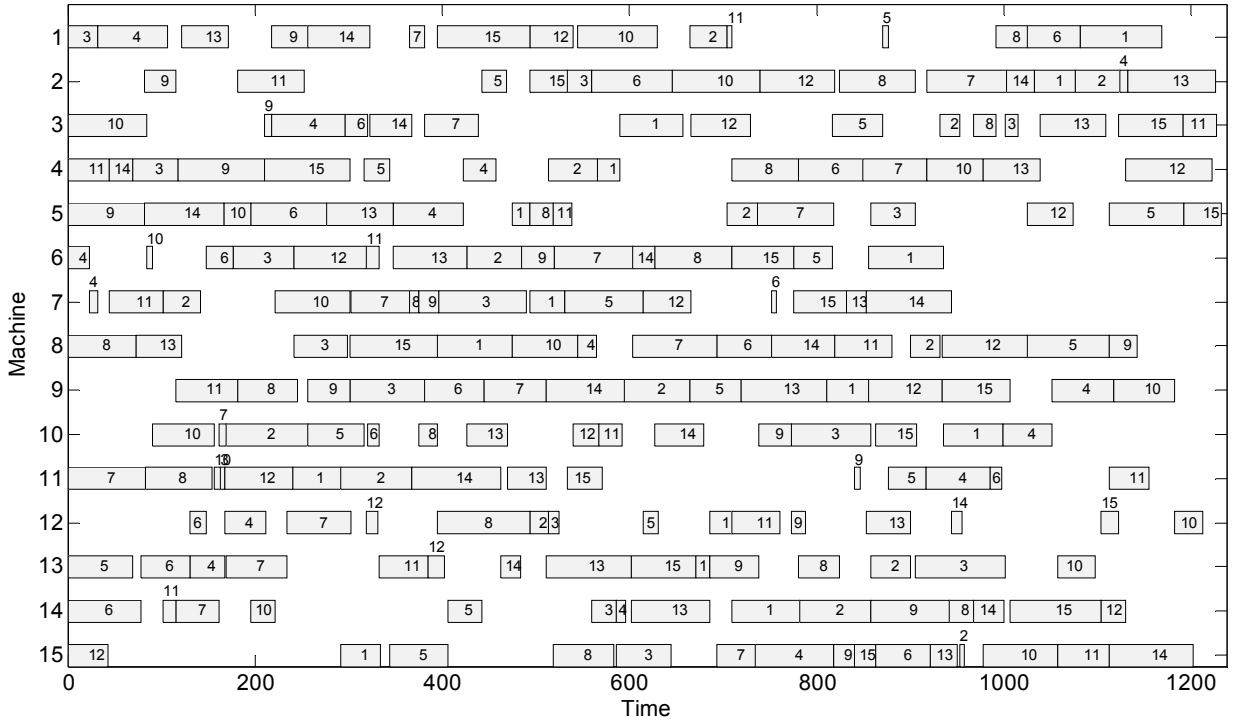

Fig. 6 Gantt chart of an optimal schedule for la39

REFERENCES

[1] T. Yamada and R. Nakano, "A genetic algorithm applicable to large-scale job-shop problems," in *Proc. of the 2nd International Conference on Parallel Problem Solving from Nature*, Amsterdam, 1992, pp.283-292.

[2] 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.

[3] E. Nowicki, C. Smutnicki C, "A fast taboo search algorithm for the job shop problem," *Management Science*, vol. 42, pp.797–813, 1996.

[4] E. Hart, P. Ross, and J. Nelson, "Producing robust schedules via an artificial immune system," in *Proc. International Conference on Evolutionary Computing (ICEC'98)*, Seoul, Korea, 1998, pp. 464-469.

[5] C.A.C. Coello, D.C. Rivera, and N.C. Cortes, "Use of an artificial immune system for job shop scheduling," *Artificial Immune Systems (ICARIS'2003), LNCS 2787*, Springer-Verlag, 2003, pp. 1-10.

[6] C.A.C. Coello, D.C. Rivera, and N.C. Cortes. "Job shop scheduling using the clonal selection principle," *Adaptive Computing in Design and Manufacture VI*, Springer-Verlag, 2004, pp. 113-124.

[7] S. Binato, W. J. Hery, D. M. Loewenstern, and M. G. C. Resende, "A GRASP for job shop scheduling," in *Essays and Surveys in Metaheuristics*. Boston, MA: Kluwer, 2001, pp. 59–80.

[8] A. S. Jain and S. Meeran "Deterministic job-shop scheduling: past, present and future," *European Journal of Operational Research*, vol. 113, pp.390–434, 1999.

[9] L.N. de Castro, F.J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Trans. Evolutionary Computation*, vol.6, pp. 239–251, June 2002.

[10] L.N. de Castro, J. Timmis, "An artificial immune network for multimodal function optimization," in *Proc. of the 2002 Congress on Evolutionary Computation*, Honolulu, 2002, pp. 699–704.

[11] E. Hart and J. Timmis, "Application areas of AIS: The past, the present and the future," *Applied Soft Computing*, vol. 8, pp.191-201, 2008.

[12] J.T. Tsai, W.H. Ho, T.K. Liu and J.H. Chou, "Improved immune algorithm for global numerical optimization and job-shop scheduling problems," *Applied Mathematics and Computation*, vol. 194, pp. 406-424, 2007.

[13] H.W. Ge, L.Sun, Y. Liang and F. Qian, "An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 38, pp. 358-368, March 2008.

[14] J. F. Goncalves , J. J. D. M. Mendes and M. G. C. Resende. "A hybrid genetic algorithm for the job shop scheduling problem," *European Journal of Operational Research*, vol. 167, p77-95, 2005.

[15] M. Chandrasekaran, P. Asokan, S. Kumanan, T. Balamurugan and S. Nickolas, "Solving job shop scheduling problems using artificial immune system," *International Journal of Advanced Manufacturing Technology* vol.31, pp.580–593, 2006.

[16] R. Cheng, M. Gen and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms-I. representation," *Computers & Industrial Engineering*, vol. 30, pp. 983-997, 1996.

[17] M. Pinedo, *Scheduling Theory, Algorithms, and System*, 2nd ed. Prentice Hall, Upper Saddle River, New Jersey, 2002, pp. 21-25.

[18] C.Y. Zhang, Y.Q. Rao and P.G Li, "An effective hybrid genetic algorithm for the job shop scheduling problem," *International Journal of Advanced Manufacturing Technology,* vol.39, pp965-974, 2008.

[19] J. E. Beasley, "OR-Library: Distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

[20] I. Sabuncuoglu and M. Bayiz, "Job shop scheduling with beam search," *European Journal of Operational Research*, vol. 118, no. 2, pp. 390–412, 1999.

[21] W. P. W. Nuijten and E. H. L. Aarts, "Computational study of constraint satisfaction for multiple capacitated job shop scheduling," *European Journal of Operational Research*, vol. 90, no. 2, pp. 269–284, 1996.

[22] J. Adams, E. Balas, D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol.34, pp391-401, 1988.