A Two-Level Genetic Algorithm for Scheduling in Assembly Islands with Fixed-Position Layouts

Wei Qin[a,1] and George Q. Huang[b]

[a] Department of Industrial and Manufacturing Systems Engineering, the University of Hong Kong.

Abstract. This paper focuses on the scheduling problem in assembly islands environment with fixed layouts. A fixed-position assembly line is always used when products (e.g., ships and planes) are too fragile, large or heavy to move. In such configuration, products normally remain in one location for its entire manufacturing period while machines, materials and workers are moved to an assembly site called an assembly island. Such layouts can afford necessary flexibility and competitive operational efficiency for products of modest variety and production volumes. However, the high dynamics of material, equipment and manpower flows in assembly islands make the production scheduling quite difficult. The authors give the definition and mathematical model for the scheduling problem. A two-level genetic algorithm is used to obtain a near optimal solution to minimize the makespan. Experimental results show that this algorithm is more effective in airline or ship industrial manufactures than in other machine or tool final assembly companies. It also can be found that some function of the number of jobs and the number of islands is the most important factor to the time of scheduling.

Keywords. Scheduling, two-level GA, assembly islands.

1 Introduction

Fixed-position layouts are normally used in One-of-a-Kind Production (OKP) industry, when products are too fragile, large, bulky, or heavy to move, such as in some aircraft and ship manufacturing factories, container manufacturing companies and etc (Trostmann et al. [1]). Among 80,000+ Hong Kong manufacturers in Pearl River District (PRD), many lifts and moulds manufacturers are also of this type. In such configurations, machines, material, and workers are moved to an assembly site (often called an assembly island) while products normally remain in one location for its entire manufacturing (assembly) period. Advantages of fixed-position layout include reduced movement of work items; minimized damage or cost of movement, and more continuity of the assigned work force since the item does not go from one place to another (Huang et al. [2]).

[1] Corresponding Author E-mail: qoolooq@yahoo.com

The authors in this paper focus on this scheduling problem in assembly islands. It's a new form of assembly line and the mechanism is quite different from conventional job shop or flow shop problem which make the scheduling problem in such environment much more complex than in common assembly lines. Here are some distinct characteristics as follows.

Firstly, the islands are always far away from each other, so the affection of distance is quite important here. The time spent on moving and delivering can not be omitted as in conventional assembly lines. When calculating the setup times between jobs and operations in assembly islands environment, we should take account of the distance affection.

Secondly, in assembly islands layouts, the schedule is more complex than in common job shop or flow shop problems. The schedule here should contain a part that which island the job is to be operated on. And there is also a buffer restriction in assembly islands, which makes the scheduling problem like a blocking or no-buffer scheduling question with some differences.

The remainder of this paper is organized as follows. In section 2, we present a brief literature review of scheduling in assembly islands. In section 3, we give the description and definition of the problem. Section 4 discusses the proposed genetic algorithm scheduling problem. In section 5, we report experiments and computational results. And finally is the summary.

2 Literature Review

The scheduling problem considered in this paper can be described against the literature along following directions.

2.1 Research of Assembly Islands

A typical conventional assembly line has one (or more) stationary worker who stays at a given workstation to perform an assembly operation and, when this operation is accomplished, the part assembled is transferred to another worker at the next workstation for the next assembly operation.

Linear walking-worker assembly lines are a novel form of flexible assembly system where fitters travel along the line carrying out each assembly task at each workstation. They attempt to combine the flexibility of the workbench system with the efficiency of the conventional fixed-worker assembly line.

Wang, Owen and Mileham [3] investigated a local manufacturing company in Bath, UK, where the walking-worker assembly line has been implemented. They compared the system's performance based on the same production line operated with fixed workers or with walking workers. The results have shown that by using multiple-skilled walking workers this novel system has several advantages over the conventional fixed-worker assembly line under similar conditions.

Unfortunately, quite limited reference can be found for modeling and solving the scheduling problem in the configuration of assembly islands.

2.2 Scheduling with Setup Times

In the classical scheduling problem, the job's setup times are either included in the processing times or ignored. However, in many real-life industrial problems, the setup is not a part of processing and the required time is sequence-dependent. As surveyed in Cheng *et al*. [5], operations of a job can be presented in three phases: separable setup, processing and separable removal times. Furthermore, the flow shop problems with setup time can be put into four categories according to the sequence dependence and family setup time. Considering the parallel machines with sequence-dependent setup time, Lee and Pinedo [6] proposed a three phase heuristic for minimizing the sum tardiness. A specific industrial case of parallel machines has been studied by Luh *et al*. [7]. A near-optimal solution was obtained through an integrated solution methodology based on a combined Lagrangian relaxation, dynamic programming and heuristics.

Based on the above brief literature review, we could notice that setuptimes have been well investigated in job shop and flow shop problems, but no references can be found in assembly islands scheduling problem.

2.3 Summary

Based on the above brief literature review, several observations can be made. Firstly, on the aspect of assembly islands, quite limited reference can be found. And most of them have not yet come to the scheduling problem. Secondly, in terms of scheduling with setup times and no-buffer, there are some references. However, little of them take account of the affection of distance which is important in assembly islands configuration and jobs sequences into assembly islands.

3 Problem Definition and Formulation

3.1 Problem Definition

We start with a formal definition for the scheduling problem. In a typical assembly islands environment with fixed-position layout, there are k identical assembly sites $I_1, I_2, ..., I_k$, m types of assembly operators $A_1, A_2, ..., A_m$, l logistics workers $L_1, L_2, ..., L_l$ and n jobs $J_1, J_2, ..., J_n$. Each type of operators may have several alternates which could be a group of workers or a single operator. For example, type A_j has m_j alternatives $\left\{A_{1,j}, A_{2,j}, ..., A_{m_j,j}\right\}$. But here for simplicity, we assume that there is only one for each type of operators, e.g., $m_j = 1$ for any A_j , $j = 1, 2, ..., m$. A job $J_j, j = 1, 2, ..., n,$ consists of a chain of sets of sequential operations. For example, a job J_j contains n_j operations

$\left\{O_{1,j}, O_{2,j}, ..., O_{n_j,j}\right\}$, which means it should be processed by operators $A_1, A_2, ..., A_{n_j}$ sequentially. The operator of type A_i finished the operation $O_{i,j}$ of job J_j in assembly island $I(j)$ and then go to assembly island $I(k)$ to process the operation $O_{i,k}$ of job J_k.

Each job J_j has its specified characteristics such as its operations, due date, weight and so on. And usually the total order size is relatively small. In general case ($n > k$), when the order gets started, the first batch of jobs are sent to the assembly islands $I_1, I_2, ..., I_k$ and the operators, machines and materials according to the requirement of the first operation of these jobs are also moved to the corresponding assembly island. Here we don't take account of the case of recirculation, that is, one type of operators never processes the same job more than one time.

Each operation $O_{i,j}$ has the processing time $p_{i,j}$. The release time of operation $O_{i,j}$ is denoted by $r_{i,j}$, the start time is denoted by $s_{i,j}$ and the completion time is denoted by $c_{i,j}$. We have precedence constraints of the form $O_{i,j} \to O_{i+1,j}$. Each job can only be processed only by one machine at a time and each machine can only process one job at a time. We denote the ready (release) time of job J_i by R_i, the start time of job J_j by S_j and the completion time of job J_j by C_j.

When job J_i is finished, Job J_j is moved in the same assembly island with the setup time ST_{ij}. The setup time between operations $O_{i,j}$ and $O_{k,j}$ is denoted by st_{ikj}.

Here to establish a reference frame, the inventory warehouse is set as the origin of coordinate. The assembly islands are set a line with coordinates denoted by $CI_1, CI_2, ..., CI_k$. The coordinates of assembly operators are denoted by $CA_1, CA_2, ..., CA_m$. The coordinates of logistic operators are denoted by $CL_1, CL_2, ..., CL_l$.

3.2 Problem Formulation

In our analysis we assume that all jobs are simultaneously available at time zero and that all processing times are positive. On each assembly island, only one job can be processed at a time, which means once a job starts, it cannot be moved out from the island until all its operations are completed. And because of the space restriction, there is no buffer in each island. We also assume that there is no

preemption, i.e., once started, an operation cannot be interrupted before completion.

We also require the assembly operators to move to the next operation as soon as they finished the previous one. Suppose the assembly operator A_k has just finished the operation $O_{k,q}$, and we have the equation $t_A(k,j) = c_{k,q}$. Then we can determine that

$$s_{k,j} = \max\left\{c_{i,j} + st_{ikj}, \frac{CI_{I(q)} - CI_{I(j)}}{\upsilon_A} + c_{k,q}\right\}$$

Another important assumption is that logistic operators are enough. In this manufacturing company, the assembly working operation takes much more time than the delivering operation does. And also because the logistic operators are paid much less than assembly operators, the company can hire relatively more workers to do the delivering jobs. This means that the logistic operators can always finish the logistic task in time to match the completion time of the previous job or operation.

Actually this is one case of 4 different situations shown in Figure 1 (Qin, Huang and Chen [4]).

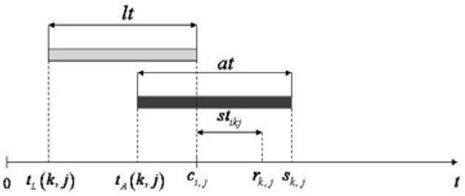

Figure 1. The logistic operators are enough.

Here the assumption indicates that

$$lt\left(CL_{L(k,j)}\left(t_L(k,j)\right), CI_{I(j)}\right) + \delta_L(k,j) = 0$$

So we get a simple equation

$$r_{k,j} = c_{i,j} + st_{ikj}$$

We consider the criterion for optimality is the makespan $C_{\max}$.

Minimize $C_{\max} = \max\{C_1, C_2, ..., C_n\}$ (1)

Subject to

$$s_{k,j} - s_{i,j} \geq p_{i,j} \text{ for all } O_{i,j} \to O_{k,j}$$

$$C_{\max} - s_{i,j} \geq p_{i,j} \text{ for all } O_{i,j}$$

$$s_{i,j} - s_{i,l} \geq p_{i,j} \text{ or } s_{i,l} - s_{i,j} \geq p_{i,j} \text{ for all } O_{i,j}, O_{i,l}$$

$$s_{i,j} \geq 0 \text{ for all } O_{i,j}$$

In this formulation the first set of constraints ensures that operation $O_{k,j}$ cannot start before operation $O_{i,j}$ is completed. The third set of constraints ensures that some ordering exists among operations of different jobs that have to be processed by the same assembly operator.

4 Proposed Two-Level Genetic Algorithm

Genetic algorithms (GA) are motivated by the theory of evolution. They date back to the early work described in Rechenberg [8], Holland [9] and Schwefel [10], also Goldberg [11] and Michalewicz [12]. They have been designed as general search strategies and optimization methods working on populations of feasible solutions.

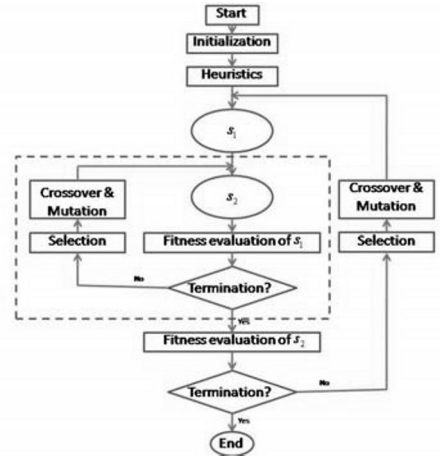

Figure 2. The flowchart of proposed GA.

In this study, Figure 2 shows the flowchart of the proposed GA approach to the scheduling problem in assembly islands. It includes two sub-problems: jobs sequences optimization and operations sequences optimization. To improve the effectiveness of the algorithm, the GA for operations sequences optimization is not executed until the optimal solution for the corresponding jobs sequence is obtained, but only executed for some generations. So in this GA algorithm, a max generation of operations sequences is pre-defined.

4.1 Solution Space

In a general scheduling problem, a schedule should include all jobs. But in our research, the situation is somewhat different. Let S be a schedule for this

$A_{nkm} \mid\mid C_{\max}$ problem. Because of the restriction of assembly islands, the schedule S should contain three parts denoted by S_1 and S_2.

S_1 is the permutation of jobs moved into the assembly islands which has the form:

$$\left\{\left(J_{I_1,1}, J_{I_1,2}, ...\right), \left(J_{I_2,1}, J_{I_2,2}, ...\right), ..., \left(J_{I_k,1}, J_{I_k,2}, ...\right)\right\}$$

S_2 is the arrangement of operations being processed by corresponding operators, which has the form:

$$\left\{\left(O_{A_1,1}, O_{A_1,2}, ...\right), \left(O_{A_2,1}, O_{A_2,2}, ...\right), ..., \left(O_{A_m,1}, O_{A_m,2}, ...\right)\right\}$$

Actually, here $\left(J_{I_i,1}, J_{I_i,2}, ...\right)$, $i = 1, 2, ..., k$ is the list of jobs to be sequentially operated on assembly island I_i. $\left(O_{A_j,1}, O_{A_j,2}, ...\right)$, $j = 1, 2, ..., m$ is the list of operations to be processed by assembly operator A_j. Encoding and decoding of this solution space would be introduced in section 4.2.

4.2 Encoding and Decoding

The design of chromosomal representation to encode the solutions has significant impact on the efficiency of algorithm (Ho *et al.* [13]). Actually, encoding is the most important part and also bottleneck in GA.

Here in this problem, normally, there are n jobs waiting for processing at time zero and only k assembly islands ($k < n$). So there are at most only k jobs can be processed simultaneously. We cannot use only one chromosome to describe the solution.

For an n-job k-island and m-operator scheduling problem, as mentioned in above section, we propose to use two chromosomes s_1 and s_2. s_1 tells the information about sequences of jobs and which assembly island they are processed on. s_2 indicates the sequences of operations. As for s_1, we adopt preference list-based representation to encode the part solution S_1. A chromosome s_1 is formed of k subchromosomes, each for one island. Each subchromosome is a string of symbols and each symbol identifies a job that has to be processed on relevant island. Once s_1 is confirmed, this problem becomes much the same as a classical job-shop scheduling problem with the differences of setup times and recirculation. We use operation-based representation to encode the part solution S_2. We named all operations for a job with the same symbol and then interpreted it according to the order of occurrence in the given chromosome. It is easy to see that any permutation of the chromosome s_2 is feasible.

Table 1. Summary of a Simple Example

Jobs	Operations Sequence	Processing times
J_1	1, 2, 3	4, 2, 2
J_2	2, 3, 1	6, 5, 6
J_3	1, 3, 2	1, 2, 7
J_4	3, 1, 2	8, 3, 5

Consider the 4-job 2-island and 3-operator scheduling problem shown in Table 1. Suppose a chromosome s_1 is given as [(1 3) (2 4)]. This indicates that J_1 and J_3 are processed on assembly island I_1 sequentially, while J_2 and J_4 are processed on assembly island I_2 sequentially. Now the problem is simplified to a 2-job 3-machine job-shop scheduling problem shown in Figure 3. We combine J_1 and J_3, J_2 and J_4, with J_{I_1} and J_{I_2} to denote them separately.

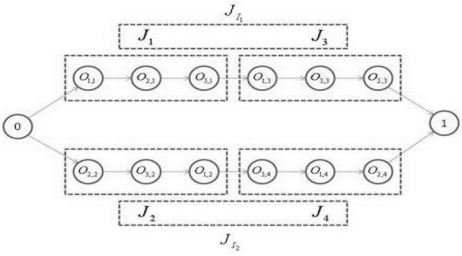

Figure 3. The disjunctive graph.

Now suppose a chromosome s_2 has the form [2 1 1 2 2 2 1 1 1 2 1 2]. Because each job has three operations, it occurs exactly six times in the chromosome. Each gene uniquely indicates an operation and can be determined according to its order of occurrence in the sequence. Let $O_{i,j}$ denote the ith operation of job J_j. The chromosome can be translated into a unique list of ordered operations of [$O_{2,2}$ $O_{1,1}$ $O_{2,1}$ $O_{3,2}$ $O_{1,2}$ $O_{3,4}$ $O_{3,1}$ $O_{1,3}$ $O_{3,3}$ $O_{1,4}$ $O_{2,3}$ $O_{2,4}$]. Operation $O_{2,2}$ has the highest priority and is scheduled first, then $O_{1,1}$, and so on. The resulting active schedule is shown in Figure 4 and Figure 5.

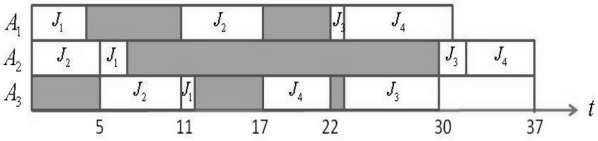

Figure 4. The active schedule.

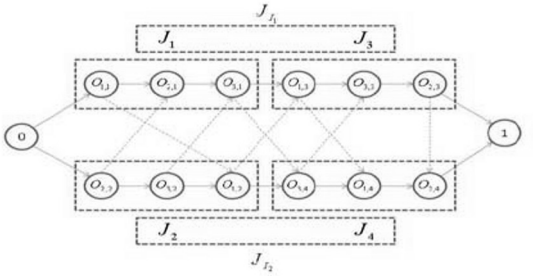

Figure 5. The disjunctive graph with the active schedule.

4.3 Fitness

The fitness is used to measure the performance of each individual. It depends on the objective. In this study, the fitness can be presented as the deviation of individual's makespan from the maximal one among all individuals.

$$fitness(x) = \max_{x \in pop}(C_{\max}(x)) + 1 - C_{\max}(x)$$

4.4 Selection

The aim of selection is to keep good individuals and eliminate bad ones. This selection is based on the fitness value of individuals. The fittest individual of current population will be chose by using the elitism selection. And then, the roulette wheel method will be used to select rest $pop - 1$ individuals. All of the pop selected individuals will form a mating pool for the crossover and mutation. We perform the roulette wheel selection in this paper.

4.5 Crossover

In general, the crossover operator is regarded as a main genetic operator and the performance of the genetic algorithms depends, to a great extent, on the performance of the crossover operator used. Conceptually, the crossover operates on two chromosomes at a time and generates offspring by combining features of both chromosomes.

In this study, we adopt the linear order crossover (LOX) operator. The crossover operator can preserve both the relative positions between genes and the absolute positions relative to the extremities of parents as much as possible. The extremities correspond to the high and low priority operations.

4.6 Mutation

It is relatively easy to make some mutation operators for permutation representation. Here we adopt shift mutation. We first choose a sublist (a job)

randomly and then shifts it to a random position of right or left from the sublist's position.

5 Computational Results

This section presents computational experiments using the above algorithm. Three research questions will be discussed: (1) How does the proposed algorithm perform when solving problems of different case? (2) How do some factors such as the number of assembly islands influence the algorithm performance, (3) What are the optimal evolutionary parameters for problems of different case and complexity? In this paper, the first two questions are successfully answered and the last one will be the future work. In the following sub-section, some test-problems are firstly presented, based on which all of the computational experiments are conducted.

5.1 Test-Problems and Experimental Results

Since in assumption the logistic operators are enough, so the solution space of scheduling problem in assembly islands grows only in proportion to (1) the number of jobs n, (2) the number of operations m (equal to operators here), (3) the number of assembly islands k. The first and the third factors determine the search space of jobs sequences. The second and the third factors determine the search space of operations sequences. The test-problems should reflect the independent influence of each factor.

There are two main types of industrial manufacturers using fixed-position assembly islands. One is the airline and ship assembly, in which there are few assembly islands but quite a lot of complex operations, i.e. $m >> k$. Another type is the machine and tool final assembly, in which there are many fixed-position assembly islands but the operations of each job are quite easy, i.e. $k >> m$. In both these two cases, n is always larger than k, but in the second case the quantity of ordered products, i.e. the number of jobs n is always very large. The test-problems will discuss these two typical and different cases.

The proposed genetic algorithm has been implemented in JAVA on a Pentium IV, 2.8GHz and 512RAM. For the first proposed question, the test-problems and results are shown in Table 2. For the second question, the test-problems and results are introduced in Table 3.

One thing need to be noticed is that during the experiments, the operations and their sequence of each job are randomly chosen first.

Table 2. Algorithm performance in two cases

	Case 1						**Case 2**					
	n	k	m	n	k	m	n	k	m	n	k	m
	10	**4**	**16**	**10**	**4**	**20**	**40**	**4**	**1**	**80**	**8**	**2**
CPU time (s)	**32**			**43**			**463**			**2054**		

Table 3. Influence of different factors to algorithm performance

	m =4, k =4					n =20, m =4					n =20, k =4				
n / k / m	4	8	12	16	20	1	2	4	8	16	1	2	4	8	16
CPU time	5	10	26	81	387	13	182	387	365	105	106	299	387	606	1532

5.2 Algorithm Performance Analysis

First in Table 2, the computational results show that the performance of algorithm effectiveness is better in case 1 than in case 2. The reason can be explained from Table 3. Figure 6, 7 and 8 tell that the relationships between the algorithm effectiveness and the three factors.

When the number of jobs n and the number of operations m increase, the used time increases sharply. But if these two parameters are fixed, the used time of GA is not an increasing function of the number of islands k. As mentioned above, n and k together determine the search space of solution S_1. Although with the increase of k, the search space of solution S_2 also increases, but the most influenced factors are n and k. When they reach a certain relation, there is a maximum of used time. This is why in case 1, the performance of proposed GA is much better. Because in case 2, the number of jobs n is quite large, which makes the search space of solution S_1 grows fast, although the limited number of operations decrease the search space of solution S_2.

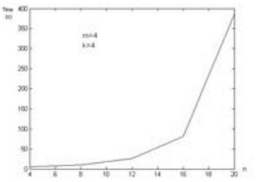

Figure 6. The relation between algorithm effectiveness and the number of jobs n.

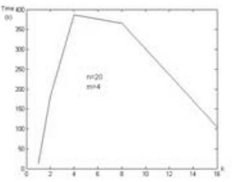

Figure 7. The relation between algorithm effectiveness and the number of islands k.

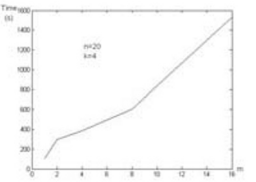

Figure 8. The relation between algorithm effectiveness and the numbers of operations m.

6 Conclusion and Future Work

This paper proposes a genetic algorithm for the scheduling problem in assembly islands with fixed-position layouts. The algorithm consists of two evolutionary processes of jobs sequences and operations sequences. They are interwoven with each other to improve the effectiveness. Experimental results show that this algorithm is more effective in airline or ship industrial manufactures than in other machine or tool final assembly companies. It also can be found that some function

of the number of jobs n and the number of islands k is the most important factor to the time of scheduling. When this function gets a certain value, the used time of algorithm will reach a maximum. The future work is to do more computationary experiments to determine this function and to find the maximum point. Besides, to take account less logistic operators into modeling is also another area for further investigation.

7 References

[1] E. Trostmann, F. Conrad, H. Holm and O. Madsen, "Cybernetic modeling and control in integrated production systems – A project review", Proceedings of the Eighth IPS Research Seminar, Denmark, pp. 213-225, 22-24 March 1993.
[2] G. Q. Huang, Y. F. Zhang, and P. Y. Jiang, "RFID-based wireless manufacturing for walking-worker assembly islands with fixed-position layouts", in Robotics and Computer-Integrated Manufacturing, vol. 23, pp. 469-477, 2007.
[3] Q. Wang, G W Owen, and A R Mileham, "Comparison between fixed-and walking-worker assembly lines", in Proc IMechE Part B: J.Engineering Manufacturing, vol. 219, pp. 845–848, 2005.
[4] P. E. Chin, G. Q. Huang and Xin Chen, "Scheduling for Assembly Islands with Fixed-Position Layouts", DET 2008, Nantes, France.
[5] T.C.Cheng, J.N.D.Gupta and G.Q.Wang, "A review of flowshop scheduling research with setup times" in Operations Management, vol. 9, no. 3, pp. 262–282, 2000.
[6] Y. H. Lee and M. Pinedo, "Scheduling jobs on parallel machines with sequence-dependent setup times", in European Journal of Operational Research, Vol. 100, pp. 464-474, 1997.
[7] P. B. Luh, L. Gou and Y. Zhang, "Job shop scheduling with group-dependent setups, finite buffers, and long time horizon", in Annals of Operations Research, 76, pp.233-259, 1998.
[8] I. Rechberg, "Opimierung technicher Systeme nach Prinzipien der biologischen Evolution", Problemate, Frommann-Holzboog, 1973.
[9] J. H. Holland, "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor, 1975.
[10] H. P. Schwefel, "Numerische Optimierung von Computer-Modellen mittels del Evolutionsstrategie",Birkhauser, Basel, 1977.
[11] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, Mass., 1989.
[12] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Springer, Berlin, 1997.
[13] N. B. Ho, J. C. Tay and E. M. K. Lai, "An effective architecture for learning and evolving flexible job-shop schedules", European Journal of Operational Research, 179, pp. 316-333, 2007.

http://www.springer.com/978-1-84882-761-5

Global Perspective for Competitive Enterprise, Economy and Ecology
Proceedings of the 16th ISPE International Conference on Concurrent Engineering
Chou, S.-Y.; Trappey, A.J.C.; Pokojski, J.; Smith, S. (Eds.)
2009, XXI, 907 p. With CD-ROM., Hardcover
ISBN: 978-1-84882-761-5