Proceedings of the 2004 IEEE
Conference on Robotics, Automation and Mechatronics
Singapore, 1-3 December, 2004

Parameter Tuning of Production Scheduling Rules by an Ant System-Embedded Genetic Algorithm

Tsung-Che Chiang[1] and Li-Chen Fu[1,2]

[1]Dept. of Computer Science and Information Engineering, National Taiwan University,

[2]Dept. of Electrical Engineering, National Taiwan University,

Taipei, Taiwan, R.O.C.

r88008@csie.ntu.edu.tw, lichen@csie.ntu.edu.tw

Abstract — **In this paper, a search algorithm is proposed for parameter tuning of production rules in job shop scheduling problems. This algorithm is developed based on the genetic algorithm, which is the core for exploration in the search space. Then an ant system is incorporated, which directs the genetic algorithm to search in the potential regions by marking potential genes for changing during reproduction. The improvement of search ability is verified by several experiments.**

I. Introduction

Production scheduling is a research field with long history due to its high complexity and practicality. Several sub-fields specific to job shops, flow shops, semiconductor manufacturing, and so on, are active both in the academia and in industry. In this paper we consider scheduling in a job shop where sequence dependent setup time and due dates of jobs are involved. The objective here is to maximize the meet-due-date rate of jobs.

The job shop scheduling problem is basically a problem of allocating resources over time to the requested jobs, and is usually reduced to a job sequencing problem. Among the approaches for job sequencing, priority rules are commonly used because of low computational requirements, ease of implementation, intuitive appeal, and flexibility to incorporate domain knowledge and expertise. Although several conventional sequencing rules [2,3,4] like SRPT (shortest remaining processing time first) rule are popular and can provide schedules with acceptable performance, researchers are still working on developing sophisticated rules for different performance criteria. For example, Lu *et al.* [5] used a class of least slack-based rules to reduce mean and variance of cycle time, and Li *et al.* [6] proposed the minimum inventory variability scheduling (MIVS) rule to achieve the same goal. Kim *et al.* [7] proposed several due-date based rules for lot release control, lot scheduling, and batch scheduling in the wafer fab.

In our previous work [1], an enhanced critical ratio-based (ECR) sequencing rule was proposed for maximizing the meet-due-date rate. This rule has four parameters, and is shown to have good performance with the default setting of parameter values. To raise its performance further, we tuned these parameters by the genetic algorithm (GA) [10], which is a well-known search algorithm. In [1], one set of parameters is followed by all stations in the plant.

Since stations may have different characteristics, the applications of the sequencing rule on different stations may require different parameter values. Tuning parameters for stations individually has the potential to improve the schedule quality; however, with the increase of number of parameters to be tuned, we need a search algorithm that can do this job more effectively, which is the motivation of this paper.

Genetic algorithms and ant colony optimization algorithm (ACO) [17] are two classes of search algorithms. The former was developed in the 70's and took the idea of biological evolutionary processes, and the latter was invented recently in the 90's and mimicked the food-seeking activities of ants. In this paper, we will take the exploration ability of GA and incorporate an ant system for intensification of the search process.

The remainder of this paper is organized as follows: Section II gives a brief review of the ECR sequencing rule, GA, and ACO. The details of the proposed search algorithm are given in Section III. Experiments and results are shown in Section IV, and concluding remarks are summarized in Section V.

II. Review of ECR, GA, and AS

A. Review of ECR sequencing rule

The ECR sequencing rule [1] is a rule derived from the concept of the critical ratio (CR) value, which is the ratio of remaining processing time and slack time of a job. The traditional CR rule gives the highest priority to the job with the largest critical ratio. Being compared to that kind of local decision, the ECR rule intends to pick a job such that the

0-7803-8645-0/04/$20.00 © 2004 IEEE

critical ratios of all jobs are kept minimal after processing it. In other words, each time when the ECR rule prioritizes a job, it calculates the sum of critical ratios of all jobs after processing the current stage of the job to be prioritized, and then select the job with the minimal sum.

As mentioned earlier, there are four parameters in the ECR rule. Parameters L and U determine a range of critical ratios [L, U], and only jobs with critical ratios falling in this range will be considered by the ECR rule. Jobs with too small critical ratios are not urgent to be processed, whereas jobs with too large critical ratios may be unlikely to meet due dates. Both kinds of jobs are not good candidates to be the next processing target and thus are filtered out before the ECR rule is applied.

The other two parameters B and D are used to calculate the critical ratios of tardy jobs. They are required because tardy jobs have negative critical ratios and will conflict with the design of the ECR rule. The details can be found in [1].

B. Review of genetic algorithms

Genetic algorithm (GA) is a search algorithm modeled after the evolution process of the nature. In the nature, organisms breed by crossover and mutation, through which the offspring inherit characteristics of parents. Based on Darwin's principle, survival of the fittest, the fitter organism has higher opportunity to survive and breed, and then the entire population evolves generation by generation. Back to GA, possible solutions are like organisms, and they "evolve" toward the optimal solution based on artificial genetic operations including crossover and so on.

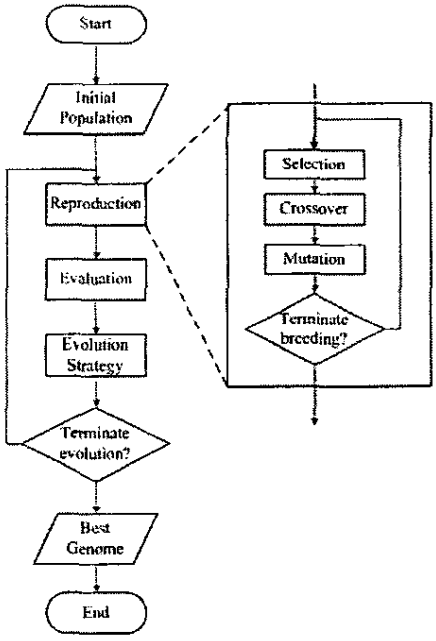

Fig. 1. The typical process flow of the genetic algorithm

To apply GA, solutions (e.g. a schedule) are first encoded to the so-called genomes in a specific representation (e.g. a job order list) so that the genetic operators can be applied conveniently. The reproduction procedure breeds offspring based on genomes at current generation. In this procedure, selection operator picks genomes to be parents, crossover operator produces offspring of the picked parents, and mutation operator adds random changes to the offspring. A fitness function is required to evaluate these genomes, and an evolution strategy decides which parents and offspring will survive to the next generation. The typical process flow of GA is depicted in Fig. 1.

Most applications of GA in production scheduling are searching for solutions in the permutation-type solution space [12, 13, 14, 16]. However, GA is also a well-recognized algorithm for searching solutions in the real number-type solution space, for example, parameter tuning in fuzzy systems [22, 23] and elevator group control systems [24]. In our previous works [1, 11], we have applied it successfully on parameter tuning of production rules, too.

C. Review of ant colony optimization algorithms

A new search algorithm called the ant colony optimization (ACO) algorithm was developed recently [17]. The concept comes from the food-seeking behaviors of ants. At the beginning, ants construct random paths to the food place. During the path construction stage, ants lay some pheromone on their paths, and an ant encountering a previously laid pheromone trail can detect it and decide with high probability to follow it. Then the amount of pheromone on the shorter path will accumulate more quickly than on the longer path, and thus attracts more ants to travel through this path and to reinforce the trail with their pheromone. The final result is that very quickly all ants will choose the shorter path.

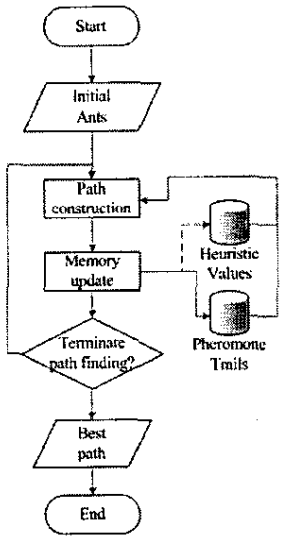

Fig. 2. The typical process flow of the ant colony optimization algorithm

In the ACO algorithm, a possible solution is represented by a path in a graph. During the stage of constructing a path, each ant moves by applying a stochastic local decision policy to determine where to go based on the information including pheromone trails and heuristic values. Once an ant builds a path, it deposits information about the goodness of the path on the pheromone trails. The heuristic values might also be updated depending on the design of the ant system. This feedback of information will direct the search of the future ants. The typical process flow of the ACO algorithm is shown in Fig. 2.

Introduction to the ACO algorithm and the applications in production scheduling can be found in [17, 18, 19, 20, 21].

III. Parameter Tuning and the AS-GA Algorithm

The main goal of this paper is to propose a search algorithm for parameter tuning of the production rules so that the schedule obtained after applying the rules can be improved further. In this section, we will describe the concept of the proposed algorithm and the details of each component.

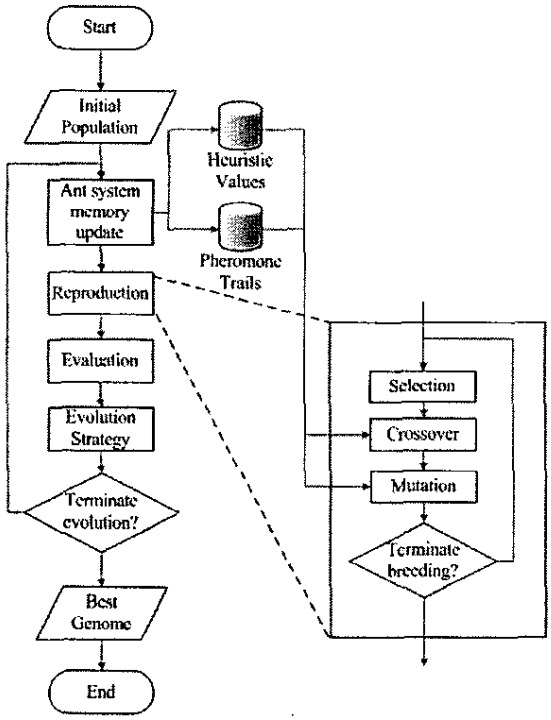

Fig. 3. The process flow of the AS-GA algorithm

The process flow of the proposed algorithm is basically the same as the flow of GA. In other words, a solution composed of sets of parameter values of rules for stations will be encoded as a genome. Then possible solutions are explored by GA through evaluation and reproduction. Since the number of parameters is large, the resulting solution space could be too huge for GA to find good solutions quickly. In order to let GA search intensively in the potential regions, an ant system is incorporated. We put an ant on each solution, and after evaluating a solution, the ant lays pheromone to mark parameters that is worth more exploration. A heuristic is also used to help marking of potential parameters. These markings will direct the reproduction procedure to produce good solutions more effectively. The process flow of our proposed algorithm is illustrated in Fig. 3.

A. Genome representation and initial population

As mentioned in subsection II.A, the ECR rule has four parameters L, U, B, and D. Here only the bounds on critical ratios, L and U, are tuned since they have stronger influence on the schedule quality than the other two in our early experiments. Therefore, given m stations in the plant, the genome is composed of $2m$ real-value numbers, which are called genes. The range of values is in [0, 1] with granularity 0.01. The value of the $(2k\text{-}1)^{th}$ gene can not be greater than the $(2k)^{th}$ gene.

The initial population of genomes is created by assigning values randomly to genes. These genomes will be evaluated immediately upon creation.

B. Evaluation and fitness function

The core of the evaluation procedure in our approach is an event-driven simulation engine. To evaluate the fitness of a genome g, we run a simulation of the plant with the ECR sequencing rule based on the parameters recorded on the genome. After simulation, the meet-due-date rate r can be calculated and this value is taken as the fitness value of the genome. In other words, we denote the above by $f(g) = r$.

C. Ant system memory update

After evaluating all genomes in a population, the heuristic values and pheromone trails in the ant system will be updated.

Heuristic value updating:

After evaluating the entire population, the heuristic values will be updated according to the best b genomes with the highest fitness found so far during the search process. The variable b is a parameter of our algorithm. Denote the best b genomes by $g_1, g_2, \ldots g_b$, and the value of the j^{th} gene of genome g_i by g_{ij}. A heuristic value η_j is associated with the j^{th} position in the genome structure, and it is set by

$$\eta_j = h(\{g_{ij}\} | i=1, 2, \ldots b)$$

where h is the function for calculating the standard deviation.

A high value of η_j means that values of the j^{th} genes of the best b genomes are very different. In our opinion, a position with higher η value is worth more exploration than one with lower η value since these good genomes do not have a conclusion of what value the gene at this position should be. When a position has a low η value, it implies that good genomes have close values at this position. In other words, the

value at this position has already converged to a small range, which is agreed by all good genomes.

Pheromone trail updating:

Similar to heuristic values, there is a pheromone trail τ_j associated with the j^{th} position in the genome structure. Before ants lay down pheromone, a pheromone evaporation procedure takes place first. It is the process by means of which the pheromone trail intensity decreases over time. Pheromone evaporation is needed to avoid a too rapid convergence of the search algorithm. It implements a useful form of forgetting, favoring the exploration of new areas of the search space. Pheromone evaporation is usually implemented by

$$\tau_j = (1-\rho)\tau_j$$

where $\rho \in (0,1]$ is the evaporation rate.

After the reproduction procedure, each parent g_p will produce an offspring $g_{o(p)}$. Based on the fitness values of parents and offspring, pheromone trails are updated by

$$\tau_j = \begin{cases} \tau_j + \dfrac{[f(g_{o(p)}) - f(g_p)]}{f(g_p)}, & \text{if } f(g_{o(p)}) > f(g_p) \text{ and } g_{pj} \neq g_{o(p)j} \\ \tau_j & \text{, otherwise} \end{cases}$$

where f is the fitness function.

Using this formula, we accumulate pheromone trails at positions where changes of gene values result in better genomes so that changes of values at these potential positions will happen with high probability in future reproduction.

D. Reproduction

There are three components in the reproduction procedure: selection operator, which chooses the genome to be a parent, and crossover and mutation operators, which produce an offspring from a parent.

Selection:

Each genome will be selected exactly once as a parent.

Crossover and mutation:

Assume the population contains n genomes, denoted by g_1, g_2, ... g_n. To produce an offspring, a parent genome g_p first duplicate itself to g_{n+p}. Then the value of each gene $g_{(n+p)j}$ will change with the probability

$$a_j = \alpha \frac{\eta_j}{2\sum_{j=1}^{2m}\eta_j} + (1-\alpha)\frac{\tau_j}{2\sum_{j=1}^{2m}\tau_j}$$

where α is a parameter that controls the relative weight of heuristic values and pheromone trails, and $2m$ is the number of genes in the genome structure. Taking an example, suppose α is 0.5, and η_j and τ_j are equal to the mean of η values and τ values respectively, then $g_{o(p)j}$ will change its value with probability $a_j = 0.5$.

When a value of gene g_{oj} is to be changed, it will change by crossover or mutation with equal probability. If the crossover operator is chosen, all genomes g_1, g_2, ... g_n have the same opportunity to donate their own genes. On the other hand, if the mutation operator is chosen, the value g_{oj} will be set by a random value in the corresponding range.

$$g_{oj} = \begin{cases} crossover: g_{pj},\ p \text{ is a random integer value in } [1,n] \\ mutation: \begin{cases} \text{a random real value in } [0, g_{o(j+1)}], \text{ if } j = 2k \\ \text{a random real value in } [g_{o(j-1)}, 1], \text{ if } j = 2k+1 \end{cases} \end{cases}$$

E. Evolution strategy

The best n genomes of all $2n$ genomes consisting of n parent genomes and n offspring genomes will survive to the next generation.

IV. Experiments and Results

A. Generation of test instances

There are three data sets in our experiments. Each data set contains ten test instances. For test instances in data set 1, there are 10 different routes, each consisting of 8 operations. The capabilities to perform these 80 kinds of operations are uniformly distributed to 25 stations. For each operation, only one station has the capability to perform it. The processing time of each operation is uniformly distributed over the interval [0.5, 5] with granularity 0.5. The sequence dependent setup time is uniformly distributed over the interval [0.5, 2] with granularity 0.5. The due date of each job is $d \times p$ where p is theoretical total processing time and d is a value uniformly distributed over the interval [4, 9] with granularity 0.1. The number of jobs is 100, and the routes of jobs are selected randomly.

Test instances in the other two data sets generally have the same specification as data set 1 except that data set 2 has d in [3, 8] and data set 3 has d in [5, 10]. We use these three data sets to represent plants with moderate, strict, and loose restriction of due dates.

B. Two benchmark search algorithms

Two benchmark search algorithms are compared with the proposed AS-GA algorithm. The first one is the random search algorithm, which is adopted here for showing the difficulty of test instances in terms of searching for good parameter values of the ECR rule. The second one is a conventional GA, which follows the typical process flow of GA, but in lack of the ant system, unlike the AS-GA algorithm. Hereafter, these two benchmark algorithms will be denoted by RS and GA.

C. Algorithm parameters

Taking into account the limitation of computation time in practical use, totally two hundreds of solutions are explored in

each run of the search algorithm. This takes about three minutes on a PC with 1G-Hz CPU and 256 MB RAM. For GA and AS-GA, the population size is ten, and the generation number is twenty.

To determine the values of crossover rate P_c and mutation rate P_m of GA and the values of evaporation rate ρ and relative weight α of AS-GA, we tested several alternatives for each algorithm with test instances in data set 1. We tested the combinations of {0.2, 0.4, 0.6}×{0.05, 0.1, 0.15, 0.2} for GA and the combinations of {0.5, 0.7, 0.9}×{0.4, 0.6, 0.8} for AS-GA. The best combination of parameter for GA is {0.6, 0.15} and the best one for AS-GA is {0.5, 0.6}. The value of b in AS-GA is set five.

D. Experimental results

In this subsection, we summarize the results after running three search algorithms over test instances in three data sets. Each search algorithm is run ten times for each instance. Let r_{ij} denote the meet-due-date rate obtained at the j^{th} run over test instance i, three values are summarized for each data set:

$$\max = \frac{\sum_{i=1}^{10} \max_{j=1,\ldots,10} \{r_{ij}\}}{10}, \quad \min = \frac{\sum_{i=1}^{10} \min_{j=1,\ldots,10} \{r_{ij}\}}{10}$$

$$avg = \frac{\sum_{i=1}^{10} avg_{j=1,\ldots,10} \{r_{ij}\}}{10}$$

They are used to judge the performance of these three search algorithms in the best, worst, and average case.

Two additional values are provided as references. The first one is the average meet-due-date rates obtained by the ECR rule with the default setting. The values of parameters L, U, B, and D are 0, 1, 1, and 0. The second one is provided to see whether using different parameters for stations is more beneficial than using identical parameters. By using identical parameters on all stations, we tested all possible combinations of $[L, U]$ from [0, 0.01], [0, 0.02] to [0.99, 1], and the highest meet-due-date rate among these 5050 alternatives is recorded. Then the average of these highest meet-due-date rates over ten instances is calculated. In the following figures, these two additional values are denoted by ECR and *BruteForce*.

From Fig. 4 – 6, we can see that AS-GA performs the best among three search algorithms either in the best, worst, or average case. The difference is larger in the data set with stricter due dates. After parameter tuning, AS-GA can always improve the meet-due-date rates even in the worst case. However, RS could generate schedules worse than those obtained by ECR when due dates are moderate and loose, and GA also has that bad behavior when due dates are loose. It reveals that lack of an effective search algorithm may make searching efforts meaningless.

Comparing average meet-due-date rates obtained by *BruteForce* and by AS-GA in the best case, we can see the potential benefits by changing from using identical parameters on all stations to using different parameters on stations individually. For each test instance i, define

F_i: the highest meet-due-date rate obtained after bruteforce search.
G_i: the highest meet-due-date rate obtained by GA in ten runs.
Si: the highest meet-due-date rate obtained by AS-GA in ten runs.
$N(X, u, v)$: number of test instances with $X - F_i$ falling in $[u, v]$.

A summary of N is shown in Table 1.

From Table 1, we see that it is possible to find better schedules by using different parameters for stations. AS-GA can find better schedules for eighteen test instances and GA can find for ten ones. Once again, the proposed AS-GA shows it superiority to the conventional GA.

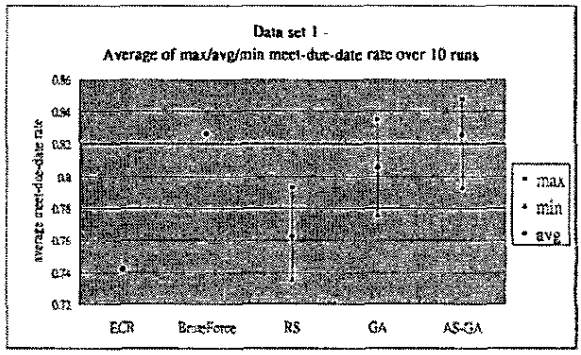

Fig. 4. Average meet-due-date rates of data set 1 (moderate due dates)

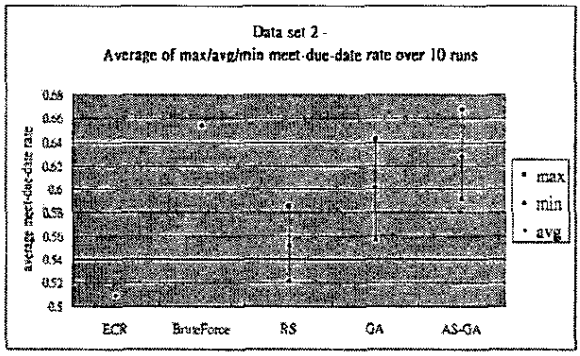

Fig. 5. Average meet-due-date rates of data set 2 (strict due dates)

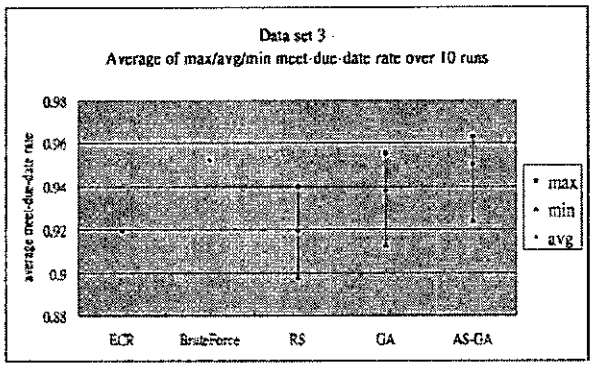

Fig. 6. Average meet-due-date rates of data set 3 (loose due dates)

Table 1. BruteForce vs. best-case GA and AS-GA

	$X = G_i$	$X = S_i$
$N_i(X, -\infty, -0.03)$	1	0
$N_i(X, -0.02, 0)$	19	12
$N_i(X, 0.01, 0.03)$	8	12
$N_i(X, 0.04, \infty)$	2	6

V. CONCLUSIONS

Motivated by tuning a large number of parameters of production rules in job shop scheduling problems, a search algorithm combining the idea of genetic algorithms and ant colony optimization algorithms is proposed. The genetic algorithm plays the major role to search in the solution space, and an ant system is incorporated to direct the genetic algorithm to search intensivcly in the potential regions. Experimental results show that this integration successfully improves the search ability of typical genetic algorithms.

Although the proposed algorithm is used for parameter tuning of production rules in job shop scheduling problems in this paper, it can be fitted for other numeric optimization problems with its general framework. This is one of our future works. Besides, we are working on developing different updating mechanisms for heuristics and pheromone trails in the embedded ant system.

References

[1] T. C. Chiang and L. C. Fu, "Solving the FMS scheduling problem by critical ratio-based heuristics and the genetic algorithm," IEEE Proc. of International Conference on Robotics and Automation, pp. 3131 – 3136, 2004.

[2] Y. L. Chang, T. Sueyoshi and R. S. Sullivan, "Ranking dispatching rules by data envelopment analysis in a job shop environment," IIE Transactions vol. 28, pp. 631 – 642, 1996

[3] L. M. Wein, "Scheduling semiconductor wafer fabrication," IEEE Trans. on Semiconductor Manufacturing, vol. 1, no. 3 pp. 115 – 130, 1988

[4] K. C. Jeong and Y. D. Kim, "A real-time scheduling mechanism for a flexible manufacturing system: using simulation and dispatching rules," International Journal of Production Research, vol. 36, no. 9, pp. 2609 – 2626, 1998.

[5] S. C. H. Lu, D. Ramaswamy and P. R. Kumar, "Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," IEEE Trans. on Semiconductor Manufacturing, vol. 7, no. 3, pp. 374 – 388, 1994.

[6] S. Li, T. Tang, and D. W. Collins, "Minimum inventory variability schedule with applications in semiconductor fabrication, " IEEE Trans. on Semiconductor Manufacturing, vol. 9, no. 1, pp. 145 – 149, 1996

[7] Y. D. Kim, J. G. Kim, B. Choi and H. U. Kim, "Production scheduling in a semiconductor wafer fabrication facility producing multiple product types with distinct due dates," IEEE Trans. on Robotics and Automation, vol. 17, no. 5, pp. 589 – 598, 2001

[8] Y. H. Lee, K. Bhaskaran and M. Pinedo, "A heuristic to minimize the total weighted tardiness with sequence-dependent setups, " IIE Transaction, vol. 29, no. 1, pp. 45 – 52, 1997.

[9] S. C. Huang and J. T. Lin, "An interactive scheduler for a wafer probe center in semiconductor manufacturing," International Journal of Production Research, vol. 36, no. 7, pp. 1883 – 1900, 1998.

[10] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," Addison-Wesley, 1989.

[11] J. H. Chen, L. C. Fu, M. H. Lin, and A. C. Huang, "Petri-Net and GA-based approach to modeling, scheduling, and performance evaluation for wafer fabrication," IEEE Trans. on Robotics and Automation, vol.17, no.5, pp.619 – 636, October 2001.

[12] C. Bierwirth and D. C. Mattfeld, "Production scheduling and rescheduling with genetic algorithms," Evolutionary Compuation, vol.7, no.1, pp.1 – 17, 1999.

[13] J. B. Yang, "GA-based discrete dynamic programming approach for scheduling in FMS environments," IEEE Trans. on System, Man and Cybernetics, part B., vol.31, no.5, pp.824 – 835, October 2001.

[14] L. Al-Hakim, "An analogue genetic algorithm for solving job shop scheduling problems," International Journal of Production Research, vol. 39, no. 7, pp. 1537 – 1548, 2001.

[15] J. F. Goncalves, J J. M. Mendes and M. G. C. Resende, "A hybrid genetic algorithm foc the job shop scheduling problem," AT&T Labs Research Technical Report TD-5EAL6J, September, 2002.

[16] M. Sevaux and S. Dauzere-Peres, "Genetic algorithms to minimize the weighted number of late jobs on a single machine," European Journal of Operational Research, vol. 151, pp. 296 – 306, 2003.

[17] M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: optimization by a colony of cooperating agents," IEEE Trans. on Systems, Man, and Cybernetics – Part B, vol 26, no. 1, pp. 29 – 41, February, 1996.

[18] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristics," IEEE Proc. of the Congress on Evolutionary Computation, vol. 2, pp. 1470 – 1477, 1999.

[19] A. Bauer, B. Bullnheimer, R. F. Hartl and C. Strauss, "An ant colony optimization approach for the single machine total tardiness problem," IEEE Proc. of the Congress on Evolutionary Computation, vol. 2, pp. 1445 – 1450, 1999.

[20] C. Blum and M. Sampels, "Ant colony optimization for FOP shop scheduling: a case study on different pheromone representations," IEEE Proc. of the Congress on Evolutionary Computation, vol. 2, pp. 1558 – 1563, 2002.

[21] V. T'kindt, N. Monmarche, F. Tercinet and D. Laugt, "An ant colony optimization algorithm to solve a 2-machine bicriteria flow shop scheduling problem," European Journal of Operational Research vol. 142, pp. 250 – 257, 2002.

[22] K. K. Tan and K. Z. Tang, "Simulation of an evolutionary tuned fuzzy dispatching system for automated guided vehicles," Proc. of the 2000 Winter Simulation Conference, pp. 1339 – 1343, 2002.

[23] A. Shaout and P. Mculiffe, "Automatic tuning of a fuzzy batch job scheduler using a genetic algorithm," IEEE Proc. of the 18th International Conference of the North American Fuzzy Information Processing Society, pp. 10 – 12, 1999.

[24] A. Fujino, T. Tobita, K. Segawa, K. Yoneda and A. Togawa, "An elevator group control system with floor-attribute control method and system optimization using genetic algorithms," IEEE Trans. on Industrial Elevatronics, vol. 44, no.4, pp. 546 – 552, 1997.