A Depth-First Mutation-Based Genetic Algorithm for Flow Shop Scheduling Problems

Pei-Chann Chang¹,², Chen-Hao Liu², Chin-Yuan Fan²
¹ Department of Information Management, Yuan-Ze University
² Department of Industrial Engineering and Management, Yuan-Ze University

Abstract

This paper presents a novel memetic genetic algorithm (GA) for the Flow Shop Scheduling problem by combining mutation-based local search with traditional genetic algorithm. The local search is based on the depth-first mutation-based searching process and the depth, i.e., the number of total mutation within each generation is according to the number of jobs to be scheduled. In traditional GA, the optimal solution may just next to the current best one however the combination of crossover and mutation may generate individuals with the solution jumping off the optimal zones. Therefore, in this research the classical mutation is replaced by depth-first multiple mutations within each generation. The multi-mutation can provide a more completely deep searching during each generation therefore there are more chances for the evolving searching procedure to reach to the optimal zone. In addition, the SA based acceptance rate is designed to be incorporated into the searching procedure; therefore the convergence rate of the hybrid GA can be further improved. The test problems are selected from the OR library, and the computational results show that the hybrid GA has a better solution quality than simple GA and NEH heuristic.

Keyword: GA, Flow Shop Scheduling, Local Search

1. Introduction

Flow shop scheduling is one of the most well-known and well-studied production scheduling problems with strong engineering background [19]. The permutation flow shop problem with n jobs and m machines as studied by many researchers is commonly defined as follows. Each of n jobs is to be sequentially processed on machine 1,2,...,m. The processing time \( p_{ij} \) of job i on machine j is given. At any time, each machine can process at most one job and each job can be processed on at most one machine. The sequence in which the jobs are to be processed is the same for each machine. The objective widely used is to find a permutation of jobs to minimize the maximum completion time, i.e. makespan, \( C_{\text{\max}} \). Due to its significance both in theory and applications it is always an important and valuable study to develop effective optimization methods. Flow shop scheduling is a typical NP-hard combinational optimization problem [9]. Exact techniques are only applicable to small-size problems in practice. The solution quality of constructive methods such as CDS [1], NEH [21], etc. is not rather satisfied although the process is very quick. Therefore, a lot of improvement methods such as Simulated Annealing (SA) [23][24], Genetic Algorithms (GAs) [26][27], Tabu Search (TS) [11][22][29][32] can obtain fairly satisfied solutions, but they are often very time-consuming, parameter-dependent, as well as the stopping criteria are either impracticable or hard to determine. GAs is one of the most popular evolutionary computational algorithms with learning capability and has widely gained application in a variety of fields so far. Ref. [13] developed a branch and bound algorithm for the two-machine case, which was extended to the m-machine case by Ref. [1][5] derived a new machine-based lower bound for the m-machine case and conducted computational experiments where their bound was strictly better than Bansal’s in 75-95% of the time. These experiments did not show that the new bound improved the branch and bound algorithm’s performance. Note that all these papers assume unweighted flow time.
The ideas involved in Genetic Algorithms (GAs) were originally developed by Ref. [12] and his associates, and applied to topics of interest to researchers in Artificial Intelligence. Thus there have been applications to classification, pattern recognition and machine learning. However, apart from some work on the Traveling Salesman Problem (not totally successful), little has been published on applications to combinational optimization. Flowshop scheduling is one of the most well-known problems in the area of scheduling. Various approaches to this problem have been proposed since the pioneering work of Ref. [16]. GAs have been applied (Ref. [7][10][12]) to combinotorial optimization problems such as the traveling salesman problem (Ref. [14][30][31]), and scheduling problems (see for example, Ref. [8][19]). A Simulated Annealing (SA) approach to the flow shop problem was proposed by Ref. [23][24]. This approach was shown to produce high quality solutions. A GAs for flowshop scheduling was proposed by Ref. [26], which was then tested on several categories of problems with time gradients and job correlations and some hard test problems proposed by Ref. [28]. GAs was overall seen to produce results comparable to SA for the flow shop sequencing problem for most types and sizes of problems. Further, GAs was shown to perform relatively better for large problems and reaches near-optimal solutions earlier. Both GAs and SA out-perform other heuristics. Ref. [20] compared various crossover and mutation operators and show that the two point crossover and shift mutation operators are effective for this problem.

Based on the previous literature review, the famous approach, NEH will be used to find the initial solutions so as to reduce the time spending on solution convergence. Then apply to the formal GAs procedures, the optimal/near-optimal solutions will be found; it is the traditional process for the flowshop problems. This may take a lot of time for jumping out the local optimal by crossover and mutation operators. The optimal solution may just next to the current one but these two operators will generate individuals with the solution jumping off the “optimal zones”.

Therefore, the general mutation operator will be substituted by depth-first multiple mutations within each generation. In addition, the SA based acceptance probability is designed for incorporated into the searching procedure. Although more replications of mutation will increase the computational time, the convergence rate will increase. Therefore the stable state of the hybrid GAs can be achieved early. In Section 2.1, the Genetic Algorithm for Flow Shop Scheduling will be introduced clearly. Section 2.2 presents the detailed procedure of the NEH. Section 2.3 will describe the basic process of SA and the relative literatures. Section 2.4 will make a thorough inquiry about the neighborhood concept, and we will indicate why the idea of depth-first multi-mutation has to be applied in our hybrid model. The Hybrid model of this research will be illustrated in Section 4. The testing problems are selected from the OR-Library, and the numerical results will be shown in Section 5.

2. Literature Review

2.1. Genetic Algorithm for Flow Shop Scheduling

This is a typical assembly line problem where \( n \) different jobs have to be processed on \( n \) different machines. All jobs are processed on all the machines in the same order. The processing times of the jobs on machines are fixed irrespective of the order in which the processing is done. The problem is characterized by a matrix \( P_{ij} \), \( i = 1,...,n \), \( j = 1,...,m \), of processing times. Each machine processes exactly one job at a time and each job is processed on exactly one machine at a time. The problem then is to find a sequence of jobs so as to minimize the makespan, that is, the completion time of the last job in the sequence on the last machine. If \( C_j \) denotes the completion time for job \( j \), then we are trying to minimize \( \max C_j \). There are many other criterions that can be considered for optimization. We refer the reader to Ref. [3] for a detailed discussion of multi-objective scheduling using GA. For details of the flow shop and other scheduling and sequencing problems we refer the reader to Ref. [4].

The first step in applying GA to a particular problem is to convert the feasible solutions of that problem into a string type structure called chromosome. In order to find the optimal solution of a problem, a standard GA starts from a set of assumed or randomly generated solutions (chromosomes) called initial population and evolve different but better sets of solution (chromosomes) over a sequence of generations (iterations). In each generation the objective function (fitness measuring criterion) determines the suitability of each chromosome and, based on these values, some of them are selected for reproduction. For the flow shop-scheduling problem we take the fitness value of each chromosome to be the reciprocal of the makespan. The number of copies reproduced by an individual parent is expected to be directly proportional to its fitness value, thereby embodying the natural selection procedure, to some
The procedure thus selects better (highly fitted) chromosomes and the worse one are eliminated. Genetic operators such as crossover and mutation are applied to these (reproduced) chromosomes and new chromosomes (offspring) are generated. These new chromosomes constitute the next generation. These iterations continue till some termination is satisfied. A good reference for understanding how GAs works is in Ref. [10].

The standard implementation for solving a permutation flowshop problem using GA can be found in Ref. [25]. The components of a standard GA applied to solve the flowshop problem may be briefly described as follows.

### Fig. 1. The Simple Genetic Algorithm

**Step 1. Encoding**
For the flowshop scheduling problem, each gene presents a job number. And a chromosome presents the processing sequence of all jobs.

**Step 2. Generate the Initial Population**
Randomly arrange the jobs to each chromosome. These chromosomes will generate the initial population.

**Step 3. Compute the objective value**
The objective of the flow shop scheduling problems in this research is to minimize the makespan $g(s)$ (the completion time of the last machine).

#### Step 4. Compute the fitness function
The original concept of fitness is “the larger the better”, because solutions with larger fitness tend to propagate to the next generation. This paper considers the minimization of objectives; hence it contradicts the original idea of fitness. A transformation should be made to reverse the minimization to maximization. For a solution $x$, its fitness equals to $1$ minus itself. The formula is listed as the following:

$$fit(x) = 1 - g(s)$$

#### Step 5. Reproduction / Selection
The tournament selection method is applied in this research.

**Step 6. Crossover**
One-point crossover method is applied in the research.

**Step 7. Mutation**
SWAP mutation method is applied in the research.

**Step 8. Elite Strategy**
The elite strategy retains the top 10% solutions in order to keep quality solutions of each generation.

**Step 9. Replacement**
The new population generated by the previous steps updates the old population.

**Step 10. Stopping criteria**
If the number of generations equals to the maximum generation number then stop, otherwise go to step 3.

### NEH
NEH is a good heuristic for flow shop scheduling problem that was developed by Ref. [21]. NEH is easy to construct and gives good results in most flow shop problems. The procedure about NEH will be represented as Fig.2, and the steps of NEH are detailed as follows:

#### Step 1: Construct the job list by arranging the jobs with descending ordering of their processing time

#### Step 2: Find the best sequence of jobs 1 and 2

#### Step 3: Pick the next job and insert it into the best position

#### Step 4: Repeat step 3

* The solutions of NEH is 4-1-3-2

**Fig. 2. The illustration of NEH**
1. Calculate the sum of processing times for each job. Arrange the jobs in the descending order of their sums. Call this the job list, and denote the job order as \( a_1, a_2, \ldots, a_n \).

2. Select the first two jobs from the job list. Determine the minimum makespan of two sequences, first by placing job \( a_i \) in the first place and \( a_j \) in the second place and then reversing the order. Do not change the relative positions of the two jobs with respect to each other in the remaining steps of the algorithm.

3. Pick the job that is in the next position in the job list and find the best sequence by placing it in all possible positions in the partial sequence developed so far. Make sure not to change the relative positions of the jobs that are already assigned in sequence.

4. Repeat step 3 until all jobs are placed in the sequence.

3. Simulated Annealing

Simulated annealing extends basic local search by allowing moves to inferior solutions; cf. Ref. [17][6]. The basic algorithm of simulated annealing may be described as follows: Successively, a candidate move is randomly selected; this move is accepted if it leads to a solution with a better objective function value than the current solution. Otherwise the move is accepted with a probability that depends on the deterioration \( \Delta E \) of the objective function value. The probability of acceptance is usually computed as \( e^{-\frac{\Delta E}{T}} \), using a temperature \( T \) as control parameter. This temperature \( T \) is gradually reduced according to some cooling schedule, so that the probability of accepting deteriorating moves decreases in the course of the annealing process.

From a theoretical point of view, a simulated annealing process may provide convergence to an optimal solution if some conditions are met (e.g., with respect to an appropriate cooling schedule and a neighborhood which leads to a connected solution space); cf. Ref. [2][18] which give surveys on simulated annealing with theoretical results as one main topic. As convergence rates are usually too slow, in practice one typically applies some faster cooling schedule (giving up the theoretical convergence property). We follow the robust parameterization of the general simulated annealing procedure as described by Ref. [15]. \( T \) is initially high, which allows many inferior moves to be accepted, and is gradually reduced through multiplication by a parameter \( \alpha < 1 \) according to a geometric cooling schedule. At each temperature size Factor \( N \) move candidates are tested (\( N \) denotes the current neighborhood size), before \( T \) is reduced to \( \alpha \times T \). According to the general procedure of simulated annealing, the near-optimal solution will be found. The advantage of simulated annealing is judging accept the bad solution or not, the worse solution may lead to a better solution searching area, other heuristics will not adopt the worse one. It’s the reason why we would like to take the acceptance probability as the adopting decision in our hybrid model.

**Hybrid Genetic Algorithm for Flow Shop Scheduling Problem**

In this study, we add the NEH heuristic into the initialization of GA. Because the NEH can generate a suboptimal solution rapidly, a GA with elitism can insure to obtain a solution no worse than that of the heuristic. Furthermore, because the other solutions are still generated randomly, the diversity of the initial individuals can be maintained to a certain dimension.

In a Simple Genetic Algorithm (SGA), the crossover operator is mainly focus on maintaining the searching diversity, and the mutation one is concentrate on keeping the solution searching deepness. However, the mutation operator just performs a very limited local search, that’s not deep enough. Based on this, we replace the mutation operator by multi-mutation one. Although the process may need more searching times than the SGA, we can get a much better solution in fewer generations. In other words, in the same solutions’ searching amount, the hybrid GA can get an outstanding performance than the SGA.

To enhance the performance of the genetic search and to avoid premature convergence, we make some improvements to the SGA, based on the discussion in last section and the statement “hybridize where possible to compensate GA’s shortcoming” [15], and propose a hybrid heuristic. First, we incorporate the NEH heuristic into the random initialization of the GA. Because the NEH heuristic can generate a suboptimal solution rapidly, a GA with elitism can guarantee to obtain a solution no worse than that of the heuristic. Moreover, the diversity of the initial population can be maintained to a certain extent, because the other solutions are still generated randomly.

The framework of the hybrid GA will be presented as Fig.3 and the detail process of the hybrid GA is described as follows:
Fig. 3. The procedure of hybrid genetic algorithm

1. Generate an individual by NEH, and the other N-1 individuals generate randomly.
2. Calculate the fitness.
3. Determine the elite from current individuals.
4. Use tournament selection method to choose N individuals from the current individual set.
5. Replace some individuals by the elite ones.
6. Generate offspring by One-points crossover.
7. Mutate the offspring M times (SWAP). At the beginning, we will define an Accept Probability (AP). And each time, based on the AP, we will determine accept the new offspring or not. If mutate time is equal to M, we define at the beginning, go to 8.
8. Terminate the Hybrid GA when the stopping criterion is reached.

Computational Results

We applied our method to some of Taillard’s benchmark problems [28] (ta problems, in short). 20 runs were carried out for each problem with different random seeds.

In this study, all the computational simulations were executed on a PentiumIV/1.5G PC at 10 independent times. The crossover rate is 0.95, population size is 50, the mutation times, \( M = n \times m / 10 \). \( n \) is the number of job, and \( m \) is the number of machine. The best parameters in this hybrid GA were got from the following two figures, Fig.4 and Fig.5. According to the two convergence charts, we found the best accept probability is 0.9 and the best mutation rate is 0.7.
computational time than NEH, it can achieve a better solution quality. And compare with SGA, at the same searching times, the HGA is also better than SGA. That is because SGA is hardly to jump out the local optimal solution, even it runs more generations. Therefore, we conclude that the global and local search ability of HGA is better than SGA and other Heuristic.

Moreover, to explore the solution space over a wide range to obtain high efficiency and good quality, these kinds of neighbor state generators act on the current population simultaneously, where SWAP acts on half of the population, and INV and INS act on the residual two quarters of the population, respectively. Because the size of the search space is $n!$, at each temperature the metropolis sample process is repeated $n$ times to provide a good compromise between solution quality and search efficiency.

Table 1. The results of NEH, SGA and HGA

<table>
<thead>
<tr>
<th>Problem</th>
<th>$n, m$</th>
<th>$C^*$</th>
<th>NEH</th>
<th>SGA</th>
<th>HGA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$Z_{N_{EH}}$</td>
<td>Error Rate</td>
<td>$Z_{S_{GA, best}}$</td>
</tr>
<tr>
<td>Ta001</td>
<td>20,5</td>
<td>1278</td>
<td>1286</td>
<td>0.63%</td>
<td>1297</td>
</tr>
<tr>
<td>Ta006</td>
<td>20,5</td>
<td>1195</td>
<td>1228</td>
<td>2.76%</td>
<td>1210</td>
</tr>
<tr>
<td>Ta011</td>
<td>20,10</td>
<td>1582</td>
<td>1680</td>
<td>6.19%</td>
<td>1611</td>
</tr>
<tr>
<td>Ta016</td>
<td>20,10</td>
<td>1397</td>
<td>1453</td>
<td>4.01%</td>
<td>1422</td>
</tr>
<tr>
<td>Ta021</td>
<td>20,20</td>
<td>2297</td>
<td>2410</td>
<td>4.92%</td>
<td>2316</td>
</tr>
<tr>
<td>Ta026</td>
<td>20,20</td>
<td>2226</td>
<td>2349</td>
<td>5.53%</td>
<td>2242</td>
</tr>
<tr>
<td>Ta031</td>
<td>50,5</td>
<td>2724</td>
<td>2733</td>
<td>0.33%</td>
<td>2724</td>
</tr>
<tr>
<td>Ta036</td>
<td>51,5</td>
<td>2829</td>
<td>2850</td>
<td>0.74%</td>
<td>2835</td>
</tr>
<tr>
<td>Ta041</td>
<td>50,10</td>
<td>2991</td>
<td>3146</td>
<td>5.18%</td>
<td>3131</td>
</tr>
<tr>
<td>Ta046</td>
<td>50,10</td>
<td>3006</td>
<td>3178</td>
<td>5.72%</td>
<td>3135</td>
</tr>
<tr>
<td>Ta051</td>
<td>50,20</td>
<td>3850</td>
<td>4038</td>
<td>4.88%</td>
<td>4001</td>
</tr>
<tr>
<td>Ta056</td>
<td>50,20</td>
<td>3681</td>
<td>3918</td>
<td>6.44%</td>
<td>3863</td>
</tr>
<tr>
<td>Ta061</td>
<td>100,5</td>
<td>5493</td>
<td>5567</td>
<td>1.35%</td>
<td>5505</td>
</tr>
<tr>
<td>Ta066</td>
<td>100,5</td>
<td>5135</td>
<td>5139</td>
<td>0.08%</td>
<td>5146</td>
</tr>
<tr>
<td>Ta071</td>
<td>100,10</td>
<td>5770</td>
<td>5848</td>
<td>1.35%</td>
<td>5893</td>
</tr>
<tr>
<td>Ta076</td>
<td>100,10</td>
<td>5303</td>
<td>5373</td>
<td>1.32%</td>
<td>5361</td>
</tr>
<tr>
<td>Ta081</td>
<td>100,20</td>
<td>6202</td>
<td>6661</td>
<td>7.40%</td>
<td>6580</td>
</tr>
<tr>
<td>Ta086</td>
<td>100,20</td>
<td>6364</td>
<td>6761</td>
<td>6.24%</td>
<td>6689</td>
</tr>
</tbody>
</table>

The computational simulation is executed at 10 independent times and the statistical results and comparisons are summed up in Table 1, it can be seen that the hybrid heuristic is able to obtain much better solutions than NEH and the best SGA, for all the problems considered, and even its worst solution quality is superior to that of NEH and the SGA. The average relative errors of 10 independent simulations are quite small, which means that the hybrid heuristic is of certain robustness for the initial solutions, while the SGA is very susceptible to the initial solutions. We therefore conclude that the global exploration ability of the hybrid heuristic with multi-nutation operators and is improved, which demonstrates the effectiveness of the hybrid heuristic relative to the constructive heuristics and SGA.

4. Conclusions

Incorporating the NEH heuristic into the random initialization of a GA, and replacing mutation by the SA metropolis sample process with multiple neighbor state generators, this paper proposed an effective hybrid heuristic for flow shop scheduling. Its effectiveness was demonstrated by computational results based on some benchmarks. Because the number of papers introducing hybrid systems is growing, hybrid optimization technology is still topical.
In order to achieve better quality, time performance, and robustness, other advanced operators, e.g. global search mechanism, and problem-specific information can be introduced. Because of the generality and ease of implementation of the hybrid heuristic, it can be applied to many other optimization problems.

Acknowledgement
This paper was supported by the National Science Council of Taiwan under the Grant number NSC 94-2213-E-155-003 and the Basic Research Foundation of Yuan Ze University.

Reference


Authors

Pei-Chann Chang


Chen-Hao Liu

Received a B.S. degree in industrial engineering from Chung-Yuan Christian University, Taiwan, 2001, and M.S. degree in Industrial engineering from Da-Yeh University, Taiwan, 2003. Now, he is a PhD student from the department of Industrial Engineering at Yuan Ze University in Taiwan. He is interested in Production Scheduling, Applications of Soft Computing, Multi-Objective Optimization Problems and Multi-Criteria Decision Making. Now, he is a PhD student of the Industrial Engineering Dept. of Yuan-Ze University in Taiwan.

Chin-Yuan Fan

Received a B.S. degree in Management from Chinese-Culture University, Taiwan, 2001, and M.S. degree in Management from Da-Yeh University, Taiwan, 2003. He is interested in Applications of Soft Computing, Forecasting, Multi-Objective Optimization Problems and Multi-Criteria Decision Making. Now, he is a PhD student of the Industrial Engineering Dept. of Yuan-Ze University in Taiwan.