A Fast Genetic Algorithm for the Flexible Job Shop Scheduling Problem

Marcin Cwiek
Future Processing
Bojkwowska 37A
44-100 Gliwice, Poland
mcwiek@future-processing.com

Jakub Nalepa
Institute of Informatics
Silesian University of Technology
Akademicka 16
44-100 Gliwice, Poland
jakub.nalepa@polsl.pl

ABSTRACT
This paper presents a fast genetic algorithm (GA) for solving the flexible job shop scheduling problem (FJSP). The FJSP is an extension of a classical NP-hard job shop scheduling problem. Here, we combine the active schedule constructive crossover (ASCX) with the generalized order crossover (GOX). Also, we show how to divide a population of solutions in the high-low fit selection scheme in order to guide the search efficiently. An initial experimental study indicates high convergence capabilities of the proposed GA.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Heuristic methods

General Terms
Algorithms

Keywords
Genetic algorithm; randomized crossover; flexible job shop scheduling problem

1. INTRODUCTION
The job shop scheduling problem (JSP) is a classical NP-hard optimization problem. It consists in finding a schedule \( \sigma \) for completing \( n \) jobs \( J_i \), \( i \in \{1,2,\ldots, n\} \), on \( m \) machines \( M_j \), \( j \in \{1,2,\ldots, m\} \). Each job \( J_i \) is composed of \( h \) operations \( o_{i,k}^j \), \( k \in \{1,2,\ldots, h\} \), which must be executed in a predefined order, and \( o_{i,k}^j \) should be run on \( M_j \). Execution time of each \( o_{i,k}^j \) is given as \( \tau_{i,k}^j \). The objective of the JSP is to minimize the total time of completing all jobs, usually referred to as the makespan. \( \sigma \) must be feasible, i.e., the operations of each job must be executed in the defined order.

In this paper we consider an extension of the JSP, called the flexible job shop scheduling problem (FJSP). In the

FJSP, an operation \( o_{i,k}^j \) (the \( k \)-th operation of the \( i \)-th job) can be processed by more than one machine \( M_j \), \( j \in \{1,2,\ldots, m\} \).

The JSP, along with its numerous variants, are of a wide practical applicability, thus they have been extensively studied over the years. Due to the NP-hardness of the FJSP, a number of heuristic algorithms were proposed to solve it in acceptable time [1,3–5].

The paper is organized as follows. Section 2 outlines the genetic algorithm to solve the FJSP. The experimental study is reported in Section 3. Section 4 concludes the paper.

Algorithm 1 A genetic algorithm for the FJSP.

1: Generate a population of \( N \) feasible solutions;
2: done ← false; \( C \leftarrow \emptyset \);
3: while not done do
4: \( \text{Determine } N \text{ pairs } (\sigma_A, \sigma_B); \) // Pre-selection
5: \( \text{for all } (\sigma_A, \sigma_B) \) do
6: \( X \leftarrow \text{SelectCrossoverOperator}(P_A, P_G); \)
7: \( \sigma_c \leftarrow \text{Crossover}(\sigma_A, \sigma_B, X); \)
8: \( \sigma_c \leftarrow \text{Mutate}(\sigma_c, P_m); \)
9: \( \text{UpdateChildPool}(C, \sigma_c); \)
10: end for
11: \( \text{Form the next population}; \) // Post-selection
12: \( \text{done} \leftarrow \text{CheckStoppingCondition}(); \)
13: \( C \leftarrow \emptyset; \)
14: end while
15: return best solution;

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the

GECCO ’14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2602280.
to the child pool $C$ (line 9). Then, the next population is composed of the children residing in $C$, and the elitist strategy is applied (line 11). Finally, the best individual from the last population is returned (line 15).

3. EXPERIMENTAL RESULTS

The GA was implemented in C# and run on an Intel Core 2 Quad Q9300 (3 GB RAM) computer. Its parameters were set to the following values: $N = 100$, $P_m = 0.3$, $\tau = 20$ sec., where $\tau$ is the maximum execution time. The GA was tested on two benchmark tests\(^1\), namely mt10 and mt20. Each test was run 10 times for each GA configuration.

The minimum makespan obtained using the GA is presented in Tab. 1. Here, the HLF parameter $\epsilon$ was drawn from the $\epsilon$-$N$ best individuals (see $\epsilon = 0.05$) allowing for crossing them over with a larger number of other ones (i.e., $\sigma_A$ was drawn from the $\epsilon$-$N$ best individuals). Thus, the probability of improving the best solutions in the population increased. The search was guided fast towards the best regions of the search space by exploiting a small number of the best individuals.

The makespan averaged for 10 runs of each GA configuration is presented in Figs. 1-2, for mt10 and mt20, respectively. The results show that the exploitation of a small number of best individuals help converge to the solutions of the highest quality in short time. Moreover, the ASCX operator should be preferred to guide the search efficiently.

4. CONCLUSIONS AND FUTURE WORK

We presented a fast GA to solve the FJSP. The initial results confirm that the search can be guided by randomizing the proportion of the applied crossover operators. Also, we showed the influence of the high-low fit parameter on the convergence capabilities of the GA. Our ongoing research includes performing full benchmark tests of the proposed GA, and designing a memetic algorithm to solve the FJSP. Also, we aim at implementing a parallel version of the GA.

5. ACKNOWLEDGMENTS

This work has been partially supported by the European Regional Development Fund under Operational Programme Innovative Economy 2007–2013, based on the Agreement No. UDA-POIG.01.04.00-24-138/11-01.

---

\(^1\)See http://www.idsia.ch/~monaldo/fjsp.html

---

Table 1: The minimum makespan obtained using the GA for mt10 and mt20 (best shown in boldface). $\mathbb{P}_A \rightarrow \epsilon \rightarrow 0.05 \ 0.10 \ 0.15 \ 0.20 \ 0.25 \ 0.30 \ 0.35 \ 0.40 \ 0.45$

<table>
<thead>
<tr>
<th>$\mathbb{P}_A$</th>
<th>$\epsilon$</th>
<th>mt10</th>
<th>mt20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>971</td>
<td>941</td>
</tr>
<tr>
<td>0.10</td>
<td>0.05</td>
<td>971</td>
<td>941</td>
</tr>
<tr>
<td></td>
<td></td>
<td>971</td>
<td>941</td>
</tr>
<tr>
<td>0.30</td>
<td>0.10</td>
<td>993</td>
<td>974</td>
</tr>
<tr>
<td></td>
<td></td>
<td>976</td>
<td>981</td>
</tr>
<tr>
<td>0.50</td>
<td>0.15</td>
<td>993</td>
<td>974</td>
</tr>
<tr>
<td></td>
<td></td>
<td>983</td>
<td>990</td>
</tr>
<tr>
<td>0.70</td>
<td>0.20</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td></td>
<td></td>
<td>971</td>
<td>976</td>
</tr>
<tr>
<td>0.90</td>
<td>0.25</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td></td>
<td></td>
<td>973</td>
<td>973</td>
</tr>
<tr>
<td>0.10</td>
<td>0.30</td>
<td>994</td>
<td>971</td>
</tr>
<tr>
<td></td>
<td></td>
<td>976</td>
<td>981</td>
</tr>
<tr>
<td>0.30</td>
<td>0.35</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td></td>
<td></td>
<td>971</td>
<td>976</td>
</tr>
<tr>
<td>0.50</td>
<td>0.40</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td></td>
<td></td>
<td>976</td>
<td>981</td>
</tr>
<tr>
<td>0.70</td>
<td>0.45</td>
<td>971</td>
<td>971</td>
</tr>
<tr>
<td></td>
<td></td>
<td>971</td>
<td>976</td>
</tr>
</tbody>
</table>

---

5. ACKNOWLEDGMENTS

This work has been partially supported by the European Regional Development Fund under Operational Programme Innovative Economy 2007–2013, based on the Agreement No. UDA-POIG.01.04.00-24-138/11-01.

---

6. REFERENCES


