A Fast Genetic Algorithm for the Flexible Job Shop Scheduling Problem

Marcin Cwiek
Future Processing
Bojkowska 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 shob 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 σ 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 $o^k_{i,j}$, $k \in \{1, 2, \ldots, h\}$, which must be executed in a pre-defined order, and $o^k_{i,j}$ should be run on M_j. Execution time of each $o^k_{i,j}$ is given as $\tau^k_{i,j}$. The objective of the JSP is to minimize the total time of completing all jobs, usually referred to as the *makespan*. σ 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^k_i (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.

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.

Algorithm 1 A genetic algorithm for the FJSP.

```
1: Generate a population of N feasible solutions;
2: done ← false; C ← ∅;
3: while not done do
4:     Determine N pairs (σ_A, σ_B);                ▷ Pre-selection
5:     for all (σ_A, σ_B) do
6:         X ← SelectCrossoverOperator(P_A, P_G);
7:         σ_c ← Crossover(σ_A, σ_B, X);
8:         σ_c ← Mutate(σ_c, P_m);
9:         UpdateChildPool(C, σ_c);
10:    end for
11:    Form the next population;                    ▷ Post-selection
12:    done ← CheckStoppingCondition();
13:    C ← ∅;
14: end while
15: return best solution;
```

2. GENETIC ALGORITHM OUTLINE

In the proposed GA, a population of N randomly generated feasible solutions (Alg. 1, line 1) evolves in time. In this paper we aim at minimizing the makespan, which is the primary objective of the FJSP. Once the initial population is generated, N pairs of parents (σ_A,σ_B) are determined for the crossover (line 4). We utilize the high-low fit (HLF) selection which has high exploration capabilities [2]. In HLF, the population is sorted and divided into two parts. A parent σ_A is selected from the well-fitted part ($\epsilon \cdot N$ individuals, $0 < \epsilon < 1$), and σ_B from the other part. Here, ϵ affects the exploration and exploitation capabilities of the GA.

Then, the crossover operator is selected (line 6). Here, the ASCX [1] is chosen with the probability $\mathcal{P}_A$, and the GOX with the probability $\mathcal{P}_G$, where $\mathcal{P}_A + \mathcal{P}_G = 1$. The parents σ_A and σ_B are crossed-over (line 7) and the child σ_c is mutated using the scramble mutation with the probability $\mathcal{P}_m$ which is decreased in time (line 8). The child σ_c is added

Table 1: The minimum makespan obtained using the GA for mt10 and mt20 (best shown in boldface).

$\mathcal{P}_A \downarrow$	$\epsilon \rightarrow$	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45
0.10	mt10	971	971	976	993	976	1002	993	1017	1016
0.30		952	967	967	971	974	977	983	990	994
0.50		967	967	967	967	967	971	971	976	971
0.70		967	967	971	971	971	973	971	971	967
0.90		**941**	967	967	981	974	971	993	989	996
0.10	mt20	1293	1293	1317	1314	1313	1316	1318	1301	1332
0.30		1237	1251	1291	1316	1307	1282	1304	1279	1314
0.50		1199	1220	1233	1253	1261	1285	1309	1298	1320
0.70		**1195**	1200	1213	1223	1228	1238	1250	1274	1271
0.90		1233	1216	1213	1205	1205	1205	1205	1211	1230

to the child pool $\mathcal{C}$ (line 9). Then, the next population is composed of the children residing in $\mathcal{C}$, 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: $N = 100$, $\mathcal{P}_m = 0.3$, $\tau = 20$ sec., where τ 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 ϵ was subject to change, along with the probability of applying the ASCX ($\mathcal{P}_A$). The experimental results prove that the convergence capabilities of the GA are strongly influenced by ϵ and $\mathcal{P}_A$. Here, selecting parents from a relatively small set of well-fitted individuals (see $\epsilon = 0.05$) allowed for crossing them over with a larger number of other ones (i.e., σ_A was drawn from the $\epsilon \cdot N$ 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`.

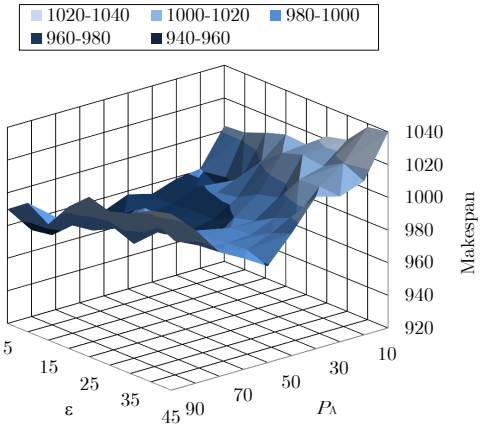

Figure 1: Average makespan for the mt10 problem (ϵ and $\mathcal{P}_A$ are given in %).

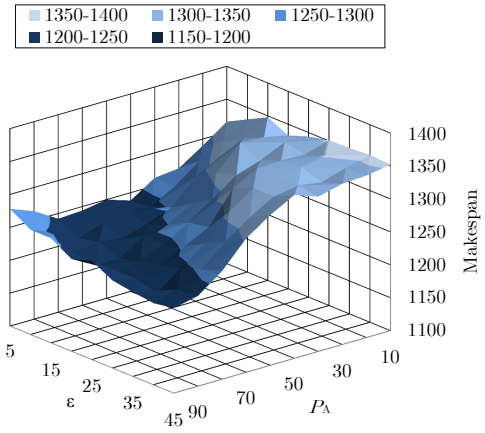

Figure 2: Average makespan for the mt20 problem (ϵ and $\mathcal{P}_A$ are given in %).

6. REFERENCES

[1] C. Dimopoulos and A.M. Zalzala. Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons. *IEEE T. Evolut. Comput.*, vol. 4, no. 2, pp. 93 – 113, 2000.

[2] M. Kawulok and J. Nalepa. Support Vector Machines Training Data Selection Using a Genetic Algorithm, in *S+SSPR 2012*, ser. LNCS, vol. 7626. Springer, 2012, pp. 557–565.

[3] M. A. Perez and F. M. Raupp. A Newton-based heuristic algorithm for multi-objective flexible job-shop scheduling problem. *J. of Intell. Manuf.*, pp. 1–8, 2014.

[4] C. R. Scrich, V. A. Armentano, and M. Laguna. Tardiness minimization in a flexible job shop: A tabu search approach. *J. of Intell. Manuf.*, vol. 15, no. 1, pp. 103–115, 2004.

[5] W. Xia and Z. Wu. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Comp. and Ind. Eng.*, vol. 48, no. 2, pp. 409 – 425, 2005.