NSGA-II Implementation Details May Influence Quality of Solutions for the Job-Shop Scheduling Problem

Maxim Buzdalov
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia
buzdalov@rain.ifmo.ru

Irina Petrova
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia
petrova@rain.ifmo.ru

Arina Buzdalova
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia
afanasyeva@rain.ifmo.ru

ABSTRACT

The helper-objective approach for solving the job-shop scheduling problem using multi-objective evolutionary algorithms is considered.

We implemented the approach from the Lochtefeld and Ciarallo paper using NSGA-II with the correct implementation of the non-dominated sorting procedure which is able to work with equal values of objectives. The experimental evaluation showed the significant improvement of solution quality.

We also report new best results for 16 out of 24 problem instances used in the considered paper.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization

General Terms

Algorithms, Performance

Keywords

job-shop, helper-objectives, auxiliary objectives, NSGA-II

1. INTRODUCTION

The job-shop scheduling problem consists of n jobs and m machines. Each job i has at most m operations. Each operation has a specified machine and processing time. The processing order of the operations is predefined. Each machine can process only one operation at time. An operation can not be interrupted once it is started. No two operations of a job can be processed simultaneously. To solve the job-shop problem means to schedule the operations on the machines to minimize the measure of scheduling. There exists several ways to measure scheduling, i.e. makespan or total flow-time. The latter is used in this paper and calculated as the sum of flowtimes for all jobs.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2602288.

Jensen [5] first applied multi-objectivization approach to the job-shop scheduling problem. Along with the target objective (the sum of all flowtimes), he introduced *helper-objectives*, which are individual flowtimes or sums of several flowtimes. The target objective and one or several helper-objectives were optimized using a multi-objective evolutionary algorithm, namely, NSGA-II [2]. The time budget for each run was divided into equal intervals, and for each interval helper-objectives were selected at random.

Later, Lochtefeld and Ciarallo [6] presented their deterministic helper-objective optimization strategy for solving the job-shop scheduling problem. In their paper, it was advised to sort the jobs by their minimum possible flowtime and join the adjacent jobs into equally sized groups to produce helper-objectives (each helper objective is the sum of flowtimes for the corresponding jobs). To be as much compatible with Jensen [5] as possible, for the sake of result comparison, they also used an implementation of NSGA-II. Their approach was shown experimentally to be quite efficient, and best known results for several problem instances were updated.

Unfortunately, neither Jensen, nor Lochtefeld and Ciarallo left any source code in the public domain to reproduce the results of experiments. When we were implementing our own approach to multi-objectivization, described in [7], we couldn't reproduce the results from [6] and had to compare our approach with our implementation of the approach from [6], which produced worse results than the original implementation.

One more difficulty comes from the multi-objective evolutionary algorithm, NSGA-II [2], used in both papers. Jensen, the author of the first paper [5], is also an author of the fast non-dominated sorting technique [4], which he seemed to use in his research. It is probably that Lochtefeld and Ciarallo also used his implementation. However, it suffers from treating equal values of objectives wrong, which was fixed by Fortin et al only very recently [3]. This means that some of the problems with reproducing the results may be related to quirks of implementations of the non-dominated sorting algorithm.

2. EXPERIMENT RESULTS

We performed 1000 runs of our implementation of the NSGA-II-based solver for the job-shop scheduling problem. The non-dominated sorting procedure is implemented as in [3]. The solver configuration, such as genetic operators and the number of iterations, and the problem instances used are the same as in [6]. We compared three helper-

Instance	Size	Best known solution	Original 1 JPH	Original 2 JPH	Original $\lfloor N/2 \rfloor$ JPH	New 1 JPH	New 2 JPH	New $\lfloor N/2 \rfloor$ JPH	New best solution
la01	10×5	4832	2.297	2.431	2.992	1.386	1.645	2.222	4832
la02	10×5	4459	3.111	3.177	2.447	2.550	2.548	2.051	4459
la06	15×5	8694	4.941	4.503	3.828	5.087	4.660	3.503	8635
la07	15×5	8174	5.425	5.245	5.091	5.770	5.709	5.249	8174
la11	20×5	14 756	5.537	5.484	5.027	5.941	5.840	5.005	14 629
la12	20×5	12 403	6.252	6.034	4.893	6.970	6.713	5.190	12 373
la16	10×10	7393	4.230	3.993	4.449	3.381	3.561	3.478	7393
la17	10×10	6555	3.555	3.696	2.617	2.366	2.649	2.088	6537
la21	15×10	12 942	4.489	4.320	4.537	3.993	3.954	3.850	12 649
la22	15×10	12 106	4.581	5.182	4.337	4.004	4.147	3.917	11 890
la26	20×10	20 234	5.290	5.480	5.924	5.515	5.621	5.268	20 013
la27	20×10	20 764	5.321	5.428	5.768	5.521	5.750	5.546	20 764
la31	30×10	39 007	5.321	5.815	6.123	6.466	6.373	6.019	39 004
la32	30×10	42 189	4.863	4.728	5.026	5.914	5.539	4.556	41 740
la36	15×15	17 073	5.052	5.348	4.943	3.963	4.257	4.318	16 793
la37	15×15	17 886	5.114	5.337	5.321	4.346	4.549	4.874	17 787
ft10	10×10	7501	6.822	6.456	7.627	5.142	5.406	5.839	7501
ft20	20×5	14 279	7.616	7.320	8.633	7.560	7.221	6.874	14 156
swv01	20×10	20 688	9.768	10.301	13.259	9.427	9.622	11.691	20 688
swv02	20×10	21 682	8.334	9.956	10.867	7.726	8.472	8.936	21 670
swv06	20×15	28 863	7.022	7.669	9.782	6.840	7.212	9.686	28 691
swv07	20×15	27 385	7.937	8.403	10.834	7.734	8.134	10.191	27 166
swv11	50×10	108 842	8.912	9.991	13.068	7.425	8.609	11.506	108 014
swv12	50×10	109 128	8.737	10.912	13.750	8.575	9.645	11.659	109 128

Table 1: Experimental results. JPH means the number of jobs per helper. In "original–new" columns, average percentages are presented, the percentage of the result X is $(X - B)/B$, where B is the previously best known solution for that problem instance. Gray cells specify which optimizer produced better average percentage, dark-gray cells show the best average percentage. In the last column, the updated best known solutions are marked gray.

objective configurations — one, two and $\lfloor N/2 \rfloor$ jobs per helper. The results are presented in Table 1.

One can see that in 54 out of 72 cases the new implementation is better. For 20 out of 24 problem instances, the best average percentage is shown by the new implementation. For 16 out of 24 problem instances we updated the best known results (the gray cells in the right column of Table 1).

Source code for experiments is available at GitHub [1].

This work was financially supported by the Government of Russian Federation, Grant 074-U01.

3. REFERENCES

[1] Source code for experiments (a part of this paper). URL: https://github.com/mbuzdalov/papers/tree/master/2014-gecco-lba-jobshop.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *Transactions on Evolutionary Computation*, 6:182–197, 2000.

[3] F.-A. Fortin, S. Grenier, and M. Parizeau. Generalizing the Improved Run-time Complexity Algorithm for Non-dominated Sorting. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, pages 615–622. ACM, 2013.

[4] M. T. Jensen. Reducing the Run-time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms. *Transactions on Evolutionary Computation*, 7(5):503–515, 2003.

[5] M. T. Jensen. Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation: Evolutionary Computation Combinatorial Optimization. *Journal of Mathematical Modelling and Algorithms*, 3(4):323–347, 2004.

[6] D. F. Lochtefeld and F. W. Ciarallo. Helper-Objective Optimization Strategies for the Job-Shop Scheduling Problem. *Applied Soft Computing*, 11(6):4161–4174, 2011.

[7] I. Petrova, A. Buzdalova, and M. Buzdalov. Improved Helper-Objective Optimization Strategy for Job-Shop Scheduling Problem. In *Proceedings of the International Conference on Machine Learning and Applications*, volume 2, pages 374–377. IEEE Computer Society, 2013.