2013 12th International Conference on Machine Learning and Applications

Improved Helper-Objective Optimization Strategy for Job-Shop Scheduling Problem

Irina Petrova, Arina Buzdalova, Maxim Buzdalov
St. Petersburg National Research University
of Information Technologies, Mechanics and Optics
49 Kronverkskiy prosp.
Saint-Petersburg, Russia, 197101
Email: petrova, afanasyeva, buzdalov@rain.ifmo.ru

***Abstract*—A single-objective optimization problem can be solved more efficiently by introducing some helper-objectives and running a multi-objective evolutionary algorithm. But what objectives should be used at each optimization stage? This paper describes a new method of adaptive helper-objectives selection in multi-objective evolutionary algorithms. The proposed method is applied to the Job-Shop scheduling problem and compared with the previously known approach, which was specially developed for the Job-Shop problem. A comparison with the previously proposed method of adaptive helper-objective selection based on reinforcement learning is performed as well.**

I. Introduction

Efficiency of a single-objective optimization problem can be increased with the helper-objective method [1]. The key idea of this method is to use some extra criteria (helper-objectives) and simultaneously optimize the primary criterion and helper-objectives [1]–[4]. It is not efficient enough to optimize all criteria simultaneously [1], [5]–[7]. So a way of selecting helper-objectives during the optimization process is needed. The paper [1] suggests that it is the most efficient to use only one helper-objective. However, no optimal strategy for choosing between helper-objectives is proposed yet [1], [2]. In this paper we propose a new way of choosing helper-objectives.

The helper-objective approach is based on multi-objective evolutionary algorithms (MOEAs) which are designed to optimize several objectives simultaneously. One of the most efficient known MOEAs [8] is NSGA-II [9]. This algorithm selects individuals according to the definition of Pareto-optimality.

Let us introduce some concepts. Suppose we have a list of criteria. One of them is the primary criterion called *target criterion*. Other criteria are treated as helper-objectives. Optimization of a helper-objective is not necessary, and helper-objectives are used only for increasing efficiency of the target criterion optimization. This means that we optimize only the target criterion and the selected helper-objective.

We apply the proposed method to the Job-Shop scheduling problem and compare the results with the existing methods of choosing helper-objectives [1]–[4].

A. Job-Shop Scheduling Problem

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 using formula $F_\Sigma = \sum_{i=1}^{n} F_i$, where F_i is the flow-time of job i — the time elapsed from the start of the execution of the first operation until the end of the execution of the last operation of the job i.

B. Previous Methods for Job-Shop

The helper-objective method can be applied to the Job-Shop scheduling problem in two different ways. In the approach from the article [1] F_i are treated as helper-objectives. The helper-objective to be optimized is chosen randomly from the set of all helper-objectives on each iteration of optimization process.

There is a method of increasing efficiency of algorithm [1], described in [2]. The helper-objectives are chosen in increasing order of minimal possible flow-time of the job corresponding to the helper-objective. Also in this article usage of the sums of flow-time of several jobs as the helper-objectives is proposed. The jobs are assigned to helper-objectives as follows. Assume that each helper-objective consists of p jobs. The first helper-objective is the sum of flow-times of p jobs with minimal possible flow-time, the second one is the sum of flow-times of the next p jobs and so on.

C. MOEA + RL: Helper-Objective Selection Method

Another method of adaptive helper-objective selection was proposed in our previous works [3], [4]. The method is called MOEA + RL, as it is based on reinforcement learning (RL) [10], [11]. In reinforcement learning, an *agent* applies *actions* to an *environment*. After applying of each action, the environment returns some representation of its state and an

978-0-7695-5144-9/13 $31.00 © 2013 IEEE
DOI 10.1109/ICMLA.2013.151

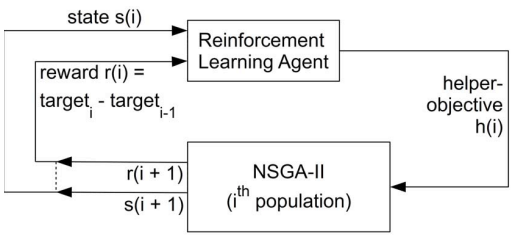

Fig. 1. Reinforcement-evolutionary interactions in the MOEA + RL method

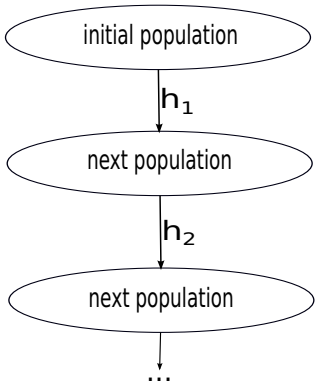

Fig. 2. The scheme of the MOEA + RL method

immediate numerical reward. The goal of a reinforcement learning algorithm is to maximize the total reward.

In the MOEA + RL method the multi-objective algorithm, MOEA, is treated as an environment. To apply an action means to select a helper-objective from the set of all helper-objectives. Once a helper-objective is selected, the next population of the MOEA is generated, the corresponding state and reward are returned and so on. The scheme of the MOEA + RL method is shown in Fig. 2, where h_i is selected with reinforcement learning. The interaction between a reinforcement learning algorithm and an evolutionary one in the MOEA + RL method is shown in Fig. 1, where i is the number of the current population. Properties of helper-objectives usually are not known in advance, so the agent should learn which objective is the most efficient one at the current optimization stage.

The reward is based on the difference of the target criterion value calculated on the best individuals in two successive generations of the MOEA. There can be different kind of state definitions. In this work we used two types of states: a single state and the *time interval state*. The latter is based on assumption that the significance of the environment changes caused by the agent increases with the number of population. So the state is logarithmically inversely proportional to the number of population and is calculated as follows: $s(t) = z - \lceil \frac{\log(n+1-t)}{\log((n+1)^{\frac{1}{z}})} \rceil$, where z is the states count and n is the total number of populations. An example of the time interval state is illustrated in Fig. 3.

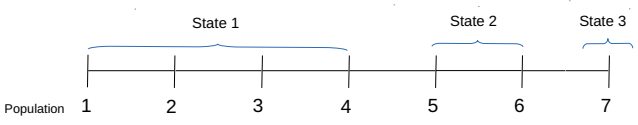

Fig. 3. Time interval state example ($z = 3, n = 7$)

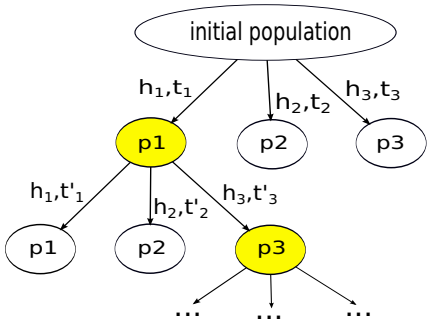

Fig. 4. The scheme of the proposed method

II. New Method Description

Let us denote helper-objectives as $H = \{h_i\}$. We optimize target criterion — total flow-time and the helper-objective which is selected from H on each iteration of the MOEA.

The scheme of the proposed method is shown in Fig. 4. Assume that there are c helper-objectives. t_i populations of the MOEA are generated for each h_i. So there are c generated populations on each iteration of the algorithm. An individual that has the best value of the target criterion is called the best individual in the population. A population which has the best individual with the best value of the target criterion is selected for the next step of the algorithm. Values of t_i are updated randomly and the next iteration is started. In the method described in Section I-C an individual that has a good value of target criterion may be lost if an inefficient helper-objective is chosen. There is no such issue in the new method, since the best individual is preserved by design.

III. Experiment Description

Helper-objectives are calculated as sums of flow-times of k jobs: $h_i = \sum_{j=k(i-1)+1}^{ik} F_j$. The jobs are assigned to the helper-objectives as proposed in the article [2]: first k jobs with minimal possible flow-time are assigned to the first helper-objective, next k jobs are assigned to the second one and so on. The number of helper-objectives is $\lfloor \frac{n}{k} \rfloor$ where n is the number of jobs.

As in the previous works [1], [2], permutations with repetitions are used for representation of an individual. For decoding individuals, Giffler-Thompson schedule builder [1], [12] is used. Generalized Order Crossover [13] and Position Based Mutation [12] are used as evolutionary operators. As in the paper [1], the mutation is always performed after crossover.

Algorithm 1 Method description

Form initial population G_0
Initialize iteration counter: $k = 0$
while (specified number of populations is not reached) **do**
  for ($h_i \in H$) **do**
    Initialize population counter: $j = 0$
    while (j is not equal to t_i) **do**
      Generate the next population g_j
      Update the population counter: $j \leftarrow j + 1$
    end while
    Count value of the target criterion of the best individual in the population g_{t_i}: $r(g_{t_i})$
  end for
  Select the next population: $G_{n+1} = g_{max},\ r(g_{max}) = \max_i r(g_{t_i})$
  Update the iteration counter: $k \leftarrow k + 1$
  Update t_i
end while

Preliminary experiments show that the best results are achieved when the number of individuals in the population is 150 and the probability of crossover is 0.8. As in the article [2] there were 20,000 fitness evaluations per run.

The NSGA-II algorithm is used in the approaches described in [1], [2] with which the proposed method has been compared, so NSGA-II is used in this work. We tried to reproduce the experiment from the article [2], but we have not achieved the same results. So we compared results obtained using the proposed method with results obtained using our implementation of the algorithm from the article [2]. Values of the NSGA-II parameters used in the experiments are shown in Table I.

TABLE I
NSGA-II PARAMETERS

Parameter	Value
Runs of the algorithm	200
Individuals in a population	150
Crossover probability	0.8
Fitness evaluations	20000

We also considered the MOEA + RL method for the same set of the Job-Shop instances using the same number of fitness evaluations per run. The Delayed Q-learning [14] algorithm was used. Preliminary experiments demonstrate that the best results are achieved using parameters shown in Table II. A single state and the time interval state described in Section I-C were used. For the time interval state, $z = 3$ was used. Two helper-objectives were considered. These values turned to be efficient during the preliminary experiment.

TABLE II
DELAYED Q-LEARNING PARAMETERS

Parameter	Description	Value
m	update period	5
γ	discount factor	0.7
ϵ	bonus reward	0.1

IV. EXPERIMENT RESULTS

Experiment results are shown in Tables III, IV, V. During the experiments, instances of the Job-Shop scheduling problem used in the papers [1], [2] were considered. The name of an instance is shown in the first column. Sizes of the instances are shown in the second column. For example, an instance of size 10×5 consists of ten jobs and five machines. The best known solution is shown in the third column. In the fourth column there are shown results of our implementation of the previously known approach [2]. This approach was designed specially for the Job-Shop problem and can not be applied for other problems. All the results are shown as average result of 200 runs in percent above the best known solution and calculated as $\frac{\text{average}-\text{best}}{\text{best}} \cdot 100\%$, where "average" is the average result, "best" is the best known solution. The dark grey background corresponds to be best result for each problem instance, while the light grey background corresponds to the second best result.

In the Table III the results obtained using the proposed method are shown. The last three columns contain results obtained using the proposed method using two, five or ten helper-objectives. For example, if the Job-Shop scheduling problem consists of ten jobs and we use two helper-objectives, each helper-objective is the sum of five jobs. For all analyzed instances the proposed method outperforms the previously known approach. The best results are achieved using two helper-objectives.

TABLE III
THE PROPOSED METHOD RESULTS

Instance	Size	Best solution	Previous method [2]	2 helpers	5 helpers	10 helpers
la01	10×5	4832	3.190	2.603	3.942	4.397
la02	10×5	4459	2.987	2.784	4.108	5.118
la16	10×10	7393	6.193	6.070	6.619	7.429
la17	10×10	6555	4.089	3.821	4.525	5.257
ft10	10×10	7501	9.216	8.936	10.448	10.859
la11	20×5	14805	9.064	8.798	11.677	13.061
la12	20×5	12484	10.364	9.729	13.441	15.728
la26	20×10	20234	10.565	10.172	11.322	12.244
la27	20×10	20844	10.261	10.117	11.135	11.922
ft20	20×5	14279	12.853	12.823	16.276	18.550
swv01	20×10	20688	20.909	20.690	23.041	24.179
swv02	20×10	21682	18.597	18.553	20.402	22.230
swv06	20×15	28863	18.026	16.920	18.545	19.490
swv07	20×15	27385	18.878	18.565	19.724	20.653

The results obtained using the previously proposed MOEA + RL method are shown in the Table IV. MOEA + RL outperformed the previously known method on 7 problem instances out of 14 ones.

The best results of each method are shown in Table V. The proposed method outperforms all considered approaches for 13 problem instances out of 14 ones. For the la-16 problem instance, the proposed method shows the second best result after the MOEA + RL method. The results obtained with the previously known method implementation are dominated for all problem instances. To sum up, the proposed method

TABLE IV
THE MOEA+RL METHOD RESULTS

Instance	Size	Best solution	Previous method [2]	Single state	Time interval state
la01	10 × 5	4832	3.190	2.787	2.972
la02	10 × 5	4459	2.987	3.221	2.953
la16	10 × 10	7393	6.193	6.288	6.007
la17	10 × 10	6555	4.089	4.084	4.226
ft10	10 × 10	7501	9.216	9.363	9.440
la11	20 × 5	14805	9.064	9.843	9.582
la12	20 × 5	12484	10.364	11.614	11.253
la26	20 × 10	20234	10.565	10.541	10.453
la27	20 × 10	20844	10.261	10.376	10.610
ft20	20 × 5	14279	12.853	14.354	13.493
swv01	20 × 10	20688	20.909	21.762	21.239
swv02	20 × 10	21682	18.597	19.025	19.124
swv06	20 × 15	28863	18.026	17.589	17.814
swv07	20 × 15	27385	18.878	18.809	19.267

of adaptive helper-objectives selection turns to be the most efficient one for the considered set of the Job-Shop problem instances.

TABLE V
COMPARING THE BEST RESULTS OF ALL METHODS

Instance	Size	Best solution	Previous method [2]	MOEA+RL	Proposed method
la01	10 × 5	4832	3.190	2.787	2.603
la02	10 × 5	4459	2.987	2.953	2.784
la16	10 × 10	7393	6.193	6.007	6.070
la17	10 × 10	6555	4.089	4.084	3.821
ft10	10 × 10	7501	9.216	9.363	8.936
la11	20 × 5	14805	9.064	9.582	8.798
la12	20 × 5	12484	10.364	11.253	9.729
la26	20 × 10	20234	10.565	10.453	10.172
la27	20 × 10	20844	10.261	10.376	10.117
ft20	20 × 5	14279	12.853	13.493	12.823
swv01	20 × 10	20688	20.909	21.239	20.690
swv02	20 × 10	21682	18.597	19.025	18.553
swv06	20 × 15	28863	18.026	17.589	16.920
swv07	20 × 15	27385	18.878	18.809	18.565

V. CONCLUSION

A new method of choosing helper-objectives during the optimization process is proposed. The achieved results outperform results of previously known method designed for the Job-Shop problem on 14 problem instances, as well as the results of the previously proposed MOEA + RL method on 13 instances. For the proposed method, usage of sums of flow-times for several jobs as helper-objectives was studied. The best results were achieved using two helper-objectives.

VI. ACKNOWLEDGMENTS

The research was supported by Ministry of Education and Science of Russian Federation in the context of Federal Program “Scientific and pedagogical personnel of innovative Russia”.

REFERENCES

[1] M. T. Jensen, “Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation: Evolutionary computation combinatorial optimization,” *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 323–347, 2004.

[2] D. F., Lochtefeld, and F. W. Ciarallo, “Helper-objective optimization strategies for the job-shop scheduling problem,” *Appl. Soft Comput.*, vol. 11, no. 6, pp. 4161–4174, 2011.

[3] M. Buzdalov and A. Buzdalova, “Adaptive selection of helper-objectives for test case generation,” in *2013 IEEE Conference on Evolutionary Computation*, vol. 1, June 20-23 2013, pp. 2245–2250.

[4] M. Buzdalov, A. Buzdalova, and I. Petrova, “Generation of Tests for Programming Challenge Tasks Using Multi-Objective Optimization,” in *GECCO (Companion)*, C. Blum and E. Alba, Eds. ACM, 2013, pp. 1655–1658.

[5] A. Buzdalova and M. Buzdalov, “Increasing efficiency of evolutionary algorithms by choosing between auxiliary fitness functions with reinforcement learning,” in *ICMLA (1)*. IEEE, 2012, pp. 150–155.

[6] ——, “Adaptive selection of helper-objectives with reinforcement learning,” in *ICMLA (2)*. IEEE, 2012, pp. 66–67.

[7] ——, “Increasing efficiency of evolutionary algorithms by choosing between auxiliary fitness functions with reinforcement learning,” in *Proceedings of the 11th International Conference on Machine Learning and Applications, ICMLA 2012*, vol. 1, 2012, pp. 150–155.

[8] C. Coello, G. Lamont, and D. Van Veldhuisen, *Evolutionary Algorithms for Solving Multi-objective Problems*, ser. Genetic and evolutionary computation series. Springer Science+Business Media, LLC, 2007.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[11] A. Gosavi, “Reinforcement learning: A tutorial survey and recent advances,” *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 178–192, 2009.

[12] M. T. Jensen, P. Dissertation, and T. Jensen, “Robust and flexible scheduling with evolutionary computation,” Tech. Rep., 2001.

[13] C. Bierwirth, “A generalized permutation approach to job shop scheduling with genetic algorithms,” *OR Spektrum*, vol. 17, pp. 87–92, 1995.

[14] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, “PAC Model-free Reinforcement Learning,” in *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, 2006, pp. 881–888.