DOI: https://doi.org/10.31410/ITEMA.2022.41

Comparing Genetic Algorithm and Variable Neighborhood Search Method for Solving Job Shop Problem

Jana Vugdelija[1]

Received: November 21, 2022
Accepted: January 24, 2023
Published: June 12, 2023

Keywords:
Job Shop;
Scheduling problem;
Genetic algorithm;
Variable neighborhood search;
Heuristics

Abstract: *Job Shop scheduling problem is one of the most complex and researched problems in the field of production planning. In this paper, two methods for solving Job Shop scheduling problem are presented and compared. The genetic algorithm and variable neighborhood search method were chosen and implemented in software for solving Job Shop problem. The paper first briefly presents Job Shop scheduling problem and then explains the development of solving software and implementation of selected solution methods. The results of using implemented genetic algorithm and variable neighborhood search method are presented on test instances with various dimensions. Solutions obtained using these two methods were put in comparison and analyzed, as well as compared with the optimal or best-known solutions in the literature.*

1. INTRODUCTION

Job Shop scheduling problem is one of the most complex and researched problems in the field of production planning. Different heuristic methods are mainly used to solve it, and in this paper variable neighborhood search method and genetic algorithm were chosen. Among other frequently used methods for solving the Job Shop problem are the ant colony algorithm (Saidi-Mehrabad et al., 2015), particle swarm optimization (Nouiri et al., 2018), artificial bee colony algorithm (Li et al., 2014), as well as many hybrid methods (Huang & Liao, 2008), (Li & Gao, 2016). The paper consists of five sections. After the introductory part, Section 2 briefly presents the Job Shop problem. In Section 3, genetic algorithm and variable neighborhood search method used to solve the Job Shop scheduling problem are described. In Section 4, the results of experiments on test instances are presented. Finally, concluding remarks are given in Section 5.

2. JOB SHOP PROBLEM

Job Shop scheduling problem represents the problem of determining the order in which products will be processed on machines, where each product can follow a different processing sequence, i.e. have a different order of operations (Pinedo, 2008). This problem belongs to the group of NP-hard problems (Garey et al., 1976). Although Job Shop may be seen as a production planning problem by setting, the concepts used to solve this problem can be applied in many other areas. For example, in the field of traffic and logistics when routing and scheduling trains or airplanes as shown by Liu and Kozan (2009), or when assigning staff and planning shifts, creating a class or lecture schedules, etc. In this paper, the basic form of the Job Shop problem (Zhang et al., 2019) is considered. Let there be n jobs, J1, J2,...Jn, that needs to be processed on m machines M1, M2,...Mm, and let each job have a known (different) sequence of processing operations that it needs to follow, with

[1] University of Belgrade, Faculty of Organisational Sciences, Jove Ilića 154, Belgrade, Serbia

each job being processed on each machine at most once. The time needed to process each product on each machine is also known, given in the form of the matrix *tij* for i=1, 2,...n and j=1, 2,...m. Each machine can process only one product at any time and each product can only be processed on one machine at any time. All jobs are of equal priority, available from the same initial moment, and it is not possible to interrupt the started processing of any product on any machine. It is necessary to find the sequence of execution of jobs on each machine in such a way that the completion time of the last processing operation on the last product is as short as possible while respecting all the aforementioned restrictions. Minimization of the total time needed to process all products and minimization of idle time on machines and others can be used as an objective function, while in this paper makespan is being considered, as previously stated.

3. GENETIC ALGORITHM AND VARIABLE NEIGHBORHOOD METHOD FOR THE JOB SHOP PROBLEM

Before implementing the selected solution methods in the software, it is necessary to determine how the solution will be represented. In the developed software, the solution is presented as a structure that contains two attributes: the schedule of execution of jobs on the machines and the time required to process all products, i.e. the objective function. The job execution schedule is written in the form of a sequence of $n \times m$ members (where n is the number of jobs and m is the number of machines). Array elements represent processing operations and have values from 1 to n, thereby indicating the job to which the operation belongs. The ordinal number of occurrence of value i in the sequence indicates the ordinal number of the i-th job operation and thus the machine on which that operation is executed. This way of recording the schedule is presented by Sevkli and Azdin (2006). The solution algorithms are presented in more detail below. Both algorithms are implemented in the programming language C# (Visual Studio environment).

3.1. Genetic Algorithm

Genetic algorithm is a heuristic first presented by Holland (1992) and is based on the idea to simulate biological process of evolution in order to find the best possible solution to various optimization problems. The very idea of applying simulated evolutionary processes in order to create a machine that learns was first mentioned by Turing (1950). The analogy between genetic algorithms and Darwin's theory of evolution is reflected in the fact that through generations the most desirable traits of individuals are favored, meaning that individuals that are better adapted, or in the case of optimization problems, have a better value of the objective function, have a higher probability of transferring their genetic material to the next generations. The methods used in genetic algorithms that simulate evolutionary processes are called genetic operators and refer to selection, crossover and mutation processes (Mattfeld, 2013). The genetic algorithm (GA) implemented to solve the Job Shop problem can be represented by the following functions:

- Creation of the initial population
- Input data: number of individuals in population, n (number of jobs), m (number of machines), order in which jobs are processed on machines, duration of processing each job on each machine
    1. Repeat until number of individuals in the population is reached
        a. Generate a random permutation of numbers from 1 to n
        b. Create a sequence of one initial solution by repeating the generated permutation m times in a row
        c. Calculate the time needed to implement the obtained schedule

2. Sort the individuals in the population into a non-decreasing sequence according to the value of the objective function

- Output data: Initial population ie. an array with a given number of individuals
- Crossover
- Input data: two parent entities, the order in which the jobs are processed on the machines, the processing time of each job on each machine
  1. Randomly select a segment of size 50% to 60% from the schedule of the first parent
  2. Rewrite the selected segment, in the same positions, in the child's schedule
  3. Make a copy of the other parent's schedule
  4. For all elements of the selected segment
     a. Remove the first occurrence of an element from a copy of the other parent's schedule
  5. In the remaining positions in the child's schedule, write the elements from the updated copy of the other parent's schedule maintaining the order
  6. Calculate the value of the objective function for the newly obtained individual
- Output data: Newly generated individual – child
- Mutation
- Input data: The unit to be mutated, the order in which the jobs are processed on the machines, the processing time of each job on each machine
  1. Randomly select two elements in the layout of the individual for mutation
  2. Replace positions of the selected elements
  3. Calculate the objective function of the mutated individual
- Output data: Altered (mutated) individual
- GA
- Input data: order in which jobs are processed on machines, duration of processing each job on each machine, number of individuals in population, n (number of jobs), m (number of machines), c (crossover rate), number of generations
  1. Create an initial population
  2. Repeat until number_of_generations
     a. Repeat until number_of_units_in_the_population
        i. Randomly select the first parent from the best c% of the population
        ii. Randomly select the first parent from the best c% of the population
        iii. Perform crossover
     b. Repeat for each individual in the new population
        i. Perform mutation
     c. Create a sequence of individuals of the old and new population sorted non-decreasingly by execution time
     d. Determine the next generation as the number_of_individuals_in_the_population of the best individuals from the sequence generated in previous step
  3. The first individual of the last generation represents the best obtained solution
  4. Present the best obtained solution on the Gantt chart
- Output data: The best solution obtained and the Gantt chart of the best solution obtained

3.2. Variable Neighborhood Search Method

Variable neighborhood search heuristics (VNS) was presented by the authors Mladenović and Hansen (1997) and since then it has been widely used in solving a large number of optimization problems. The method is based on the principle of local search, whereby the search environment

changes. After defining a set of environments and determining the initial solution, the process of improving the solution consists of a local search phase and a perturbation phase (Hansen et al., 2017). In the implemented VNS based algorithm three methods of modifying the solution were used. The adjacent solution within the first neighborhood is obtained by swapping the places of two randomly selected array elements, and while choosing a solution within the second environment, the element of the sequence and the position to which that element will be moved are randomly determined, whereby all elements in between are moved by one place. The jump (preturbation) function is defined as a multiple repetition of the method of moving to a second neighborhood solution. That is, a jump is performed by moving several randomly selected layout elements to another, also randomly selected position. The implemented algorithm can be represented by the following functions:

- Check neighborhoods for improvements
- Input data: order in which jobs are processed on machines, duration of processing each job on each machine, current solution, number of environments, maximum number of consecutive attempts without improvement
    1. Set i=1 and set the counter of consecutive attempts without improvement to 0.
    2. Repeat until maximum number of consecutive attempts without improvement is reached (Loop 1)
        a. Repeat until i is greater than the number of environments (Loop 2)
            i. Select the neighboring solution y from the i-th neighborhood of the current solution.
            ii. Calculate the objective function for y.
            iii. If that solution is better than the current one, set current solution to y, set the counter of consecutive attempts without improvement to 0 and return to the beginning of Loop 2.
            iv. Otherwise, increase i by 1 (next neighborhood) and return to the beginning of Loop 2.
        b. Increase the counter of consecutive attempts without improvement (current solution remains unchanged) and return to the beginning of Loop 1.
- Output data: Current solution
- VNS
- Input data: order in which jobs are processed on machines, duration of processing each job on each machine, n (number of jobs), m (number of machines), maximum number of jumps, maximum number of consecutive attempts without improvement
    1. Create the initial layout by repeating the sequences 1,2,3,…,n m times.
    2. Compute the objective function for the initial solution.
    3. Set the best achieved solution to the initial solution.
    4. Repeat until maximum number of jumps is reached (Loop 1)
        a. Check neighborhoods for improvements
        b. If the current solution is better than the best achieved solution, update the best achieved solution
        c. Perform a jump
        d. Increase jump counter
        e. Set the solution obtained by jumping to the current solution
        f. Compute the objective function for the newly obtained current solution
        g. Return to the beginning of Loop 1
    5. Present the best achieved solution on the Gantt chart
- Output data: Best achieved solution, Gantt chart of the best achieved solution.

4. EXPERIMENT RESULTS

For the experiment performing purposes, test instances presented by Taillard (1993) were used. The implemented solving methods were applied to groups of examples with dimensions of 15x15, 30x20 and 50x15, that is, test examples with 15 jobs and 15 machines, 30 jobs and 20 machines and with 50 jobs and 15 machines. For all examples with dimensions 15x15 and 50x15 optimal solutions are known, most of which were achieved by Brinkkötter and Brucker (1999) and Taillard (1993). Benchmark results for 30x20 examples and other test instances are presented by Shylo (2014) and updated regularly.

The stopping criterion in both implemented methods is not time-based, i.e. it is fulfilled by a certain number of generations for the genetic algorithm and the number of jumps for the variable neighborhood search method, but presented results were obtained for approximately the same time, close to 5 minutes of work per example.

Table 1 presents the results obtained by solving test examples with dimensions 15x15, and Table 2 shows the results for examples with dimensions 30x20. The best solutions obtained from 10 runs of the algorithm are shown.

Table 1. Results for 15x15 test instances

15x15	Ex. 1	Ex. 2	Ex. 3	Ex. 4	Ex. 5	Ex. 6	Ex. 7	Ex. 8	Ex. 9
GA	1282	1278	1275	1216	1276	1267	1266	1259	1351
VNS	1269	1276	1257	1220	1263	1267	1257	1258	1323
Optimal	1231	1244	1218	1175	1224	1238	1227	1217	1274
relative deviation GA	4.1%	2.7%	4.7%	3.5%	4.2%	2.3%	3.2%	3.5%	6.0%
relative deviation VNS	3.1%	2.6%	3.2%	3.8%	3.2%	2.3%	2.4%	3.4%	3.8%
Better algorithm	VNS	VNS	VNS	GA	VNS	=	VNS	VNS	VNS

Table 2. Results for 30x20 test instances

30x20	Ex. 1	Ex. 2	Ex. 3	Ex. 4	Ex. 5	Ex. 6	Ex. 7	Ex. 8	Ex. 9
GA	2374	2309	2189	2325	2260	2356	2219	2274	2281
VNS	2262	2187	2089	2209	2192	2259	2134	2184	2209
Benchmark	2006	1939	1846	1979	2000	2006	1889	1937	1963
relative deviation GA	18.3%	19.1%	18.6%	17.5%	13.0%	17.4%	17.5%	17.4%	16.2%
relative deviation VNS	12.8%	12.8%	13.2%	11.6%	9.6%	12.6%	13.0%	12.8%	12.5%
Better algorithm	VNS	VNS	VNS	VNS	VNS	VNS	VNS	VNS	VNS

As can be seen, the VNS algorithm provided better solutions for the vast majority of the first 2 sets of examples. In Table 3 are presented results achieved solving test instances with dimensions 50x15. Here, the best results out of 20 runs of the algorithms are shown.

Table 3. Results for 50x15 test instances

50x15	Ex. 1	Ex. 2	Ex. 3	Ex. 4	Ex. 5	Ex. 6	Ex. 7	Ex. 8	Ex. 9	Ex. 10
GA	3096	2932	3016	2845	2907	2968	2984	3111	3075	2877
VNS	2940	2971	2837	2884	2909	2945	3108	3061	2853	2887
Optimal	2760	2756	2717	2839	2679	2781	2943	2885	2655	2723
relative deviation GA	12.17%	6.39%	11.00%	0.21%	8.51%	6.72%	1.39%	7.83%	15.82%	5.66%
relative deviation VNS	6.52%	7.80%	4.42%	1.59%	8.59%	5.90%	5.61%	6.10%	7.46%	6.02%
Better algorithm	VNS	GA	VNS	GA	GA	VNS	GA	VNS	VNS	GA

When considering examples with greater dimensions, VNS is not as dominant compared to GA as it was for smaller test instances. In fact, both algorithms have the same number of better solutions, but the VNS has more consistently low relative deviations. For better comparison, average relative deviations from optimal or best known solutions for both algorithms for all sets of examples are shown in Table 4.

Table 4. Average relative deviations

	15x15	30x20	50x15	Total average
GA	3.8%	17.2%	7.57%	9.52%
VNS	3.1%	12.3%	6.00%	7.13%

In total scores, VNS still shows better results than GA. As hybrid versions of the genetic algorithm are often found in the literature, one assumption that would explain the results could be that the basic form of the genetic algorithm is not suitable enough for solving the Job Shop problem.

5. CONCLUSION

The paper describes solving Job Shop scheduling problem using a variable neighborhood search method and genetic algorithm. The obtained results were compared with each other. And the experiment showed that the variable neighborhood search method gave more successful results for the majority of test examples. Although the subject of the work was primarily a comparison of the two implemented methods, the obtained results were also compared with the optimal (or best known) solutions for the tested examples, where none of the methods managed to reach these values. Deviations from the optimal solutions were on average 9.52% for the genetic algorithm and 7.13% for the variable neighborhood search method. The planned direction of further research would be the development of a hybrid method that would incorporate the positive features of both algorithms with the aim of more precisely targeting the points where modification will be made (mutation or transition to a neighboring solution) and thus increase the efficiency of the algorithm.

References

Brinkkötter, W., & Brucker, P. (1999). Solving open benchmark problems for the job shop problem.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and job shops scheduling. *Mathematics of Operations Research, 1(2)*, 117–129.

Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization, 5(3)*, 423-454.

Holland, J. H. (1992), Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press

Huang, K. L., & Liao, C. J. (2008). Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & operations research, 35(4)*, 1030-1046.

Li, J. Q., Pan, Q. K., & Tasgetiren, M. F. (2014). A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Applied Mathematical Modelling*, *38*(3), 1111-1132.

Li, X., & Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, *174*, 93-110.

Liu, S. Q., Kozan, E., (2009), Scheduling trains as a blocking parallel-machine Job Shop scheduling problem, *Computers & Operations Research, 36(10)*, 2840-2852.

Mattfeld, D. C. (2013). Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling, Springer Science & Business Media.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research, 24(11)*, 1097-1100.

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(3), 603-615.

Pinedo, L. (2008). Scheduling - theory, algorithms, and systems, Prentice hall.

Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., & Mahmoodian, V. (2015). An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Computers & Industrial Engineering*, *86*, 2-13.

Sevkli, M., & Azdin, M. E. (2006), A Variable Neighbourhood Search Algorithm for Job Shop Scheduling Problems, Lecture Notes in Computer Science book series (LNCS, volume 3906).

Shylo, O. V. (2014). Job shop scheduling at Oleg V. Shylo: Personal webpage http://optimizizer.com/TA.php

Taillard, E. (1993). Benchmarks for Basic Scheduling Problems, *European Journal of Operational Research, Vol. 64, No. 2*, pp. 278-285.

Turing, A. M. (1950). Computing Machinery and Intelligence, *Mind, Vol. 59, No. 236* pp. 433–460.

Zhang, J., Ding, G., Zou, Y., Qin, S., & Fu, J. (2019). Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing, 30(4)*, 1809-1830.