Interactive Evolutionary Computation Approach to Permutation Flow Shop Scheduling Problem

Vid Keršič
University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
vid.kersic@um.si

Abstract

Artificial intelligence and its subfields have become part of our everyday lives and efficiently solve many problems that are very hard for us humans. But in some tasks, these methods struggle, while we, humans, are much better solvers with our intuition. Because of that, the question arises: why not combine intelligent methods with human skills and intuition? This paper proposes an Interactive Evolutionary Computation approach to the Permutation Flow Shop Scheduling Problem by incorporating human-in-the-loop in MAX-MIN Ant System through gamification of the problem. The analysis shows that combining the evolutionary computation approach and human-in-the-loop leads to better solutions, significantly when the complexity of the problem increases.

Keywords scheduling problems, interactive evolutionary computation, ant colony optimization, metaheuristic optimization

1 Introduction

Computers have become an essential (and sometimes the only) tool to approach complex real-world problems. But many problems are still too hard to solve, even for computers. Many of these problems are NP-hard, and their solution cannot be found by using exact algorithms [1]. Thus, many of these problems are solved with approximation and different stochastic nature-inspired population-based algorithms, where suboptimal solutions are found. However, they are still considered good enough to use in practice. Some of these problems include Travelling Salesman Problem (TSP), scheduling problems, etc. This paper focuses on the Permutation Flow Shop Scheduling Problem (PFSP), one of the scheduling problems.

PFSP is the scheduling problem where there are m machines and n jobs [2]. Each job consists of m operations, where i-th operation must be executed on the i-th machine. The order of jobs must be the same on all machines, and each machine can perform only a single operation at a time. Each operation has a specified execution time, and the goal is to minimize total job execution time, called makespan. The problem was proven to be NP-hard when $m \geq 3$, which means no efficient algorithm exists [3].

Different approximation approaches were presented in the last decades and shown to be effective in finding suboptimal solutions. Firstly, different approximation algorithms were introduced to reduce time to solve the problems and find suitable solutions [4]. Afterwards, researchers applied and adjusted different optimization algorithms, especially nature-inspired and metaheuristic optimization algorithms, to improve results in shorter time execution [5].

In the last years, a new approach emerged to solve these problems. Inspired by human intuition of problem-solving, the Interactive Evolutionary Computation (IEC) approach combines metaheuristic optimization algorithms, e.g., nature-inspired algorithms, with human-in-the-loop through some process, e.g., gamification of the problem [6]. This paper proposes an IEC approach based on the MAX-MIN Ant System (MMAS), one of the Ant Colony Optimization (ACO) algorithms, to PFSP through gamification of the problem.

The structure of the paper is as follows: Section 2 provides an overview of related work. Section 3 describes the IEC approach to PFSP. Section 4 describes the experiment, shows the results, and provides an analysis of the results. Section 5 concludes the paper.

2 Related Work

Researchers started solving PFSP almost 70 years ago, when Johnson proposed the exact algorithm, but it was restricted to two machines [2]. Many exact algorithms with different strategies were proposed in the following decades, such as the branch-and-bound algorithm [7]. Despite that, the exact algorithms could not scale to bigger instances of the problem.

To overcome the NP-hard time complexity of the problem, researchers started to develop approximation and heuristic algorithms, where different approaches and heuristics were considered, such as [8]. Methods based on the Nawaz-Enscore-Ham (NEH) heuristic proved to be one of the best performers to the problem [4]. NEH-based methods consist of two steps: first, order the jobs based on some characteristic and then insert them one by one. But the approximation algorithms still have several drawbacks, such as being dependent on certain characteristics of the specific setting of the problem and still being too slow.

Metaheuristic optimization algorithms were also consid-

DOI https://doi.org/10.18690/978-961-286-516-0.8
ISBN 978-961-286-516-0

ered for PSFP, and they achieved more favorable results than approximation algorithms, especially time-wise. Many algorithms were applied to the problem, such as Simulated Annealing, Differential Evolution, and many nature-inspired algorithms, such as ACO [9, 5]. In many benchmarks, metaheuristic algorithms achieve comparable or even better results than approximation algorithms [10].

As mentioned in the introduction, researchers tried to incorporate human intuition into metaheuristic algorithms, which proved beneficial [11]. Authors included human-in-the-loop to TSP through gamification, where a user affects pheromone level in MMAS algorithm when playing a game [6]. Several other researchers also focused on gamification of different NP-hard problems like Bin-packing problem, Job Shop Scheduling, and Open Shop Scheduling [12, 13]. Based on the related work, the proposed approach incorporates human intuition to solve the PFSP problem through the gamification process where a user affects the pheromone trail in the MMAS algorithm.

3 IEC Approach to PFSP

The goal of PFSP is to find a permutation of n jobs that minimize the total makespan. Thus, the optimization task for the problem is defined as:

$$\min C(\pi) \tag{1}$$

where π is a permutation of n jobs and C is the cost function that calculates makespan.

Based on the techniques and approaches discussed in the previous section, the solution consists of three modules[1]: video game, back-end server, and MMAS algorithm. The video game was implemented in Unity, while the back-end server and MMAS algorithm, which runs on the back-end, are written in Python.

The video game contains blocks (jobs) and a board constructed from several rows (machines). The user tries to move and stack blocks together while complying with the limitations of the problem. For each game, a new instance of the MMAS algorithm starts. Each movement performed by the user is sent to the back-end server, where the MMAS algorithm runs in iterations. For each block placement, one iteration of the algorithm is launched. Pheromone of the ants is presented as 2D matrix $\mathbf{A}$ of size $N \times N$, where N is the number of jobs. Matrix's element $\mathbf{A}_{i,j}$ is the pheromone level of positioning job i on the position j. The user's placement of blocks affects the pheromone level of the algorithm; in the proposed approach, the value is multiplied by a scalar. This change of the pheromone level is the human-in-the-loop part. After each user's movement, the solution for the current problem is recommended to the user based on the outcome and current state of the MMAS algorithm (of course, users can freely decide which move they will make next). The player is also encouraged to perform well and achieve the highest score as fast as possible since the game contains a time counter. The final game score is calculated based on the combination of makespan and the time duration of the game. The algorithm of the MMAS-IEC approach is shown in Algorithm 1.

```
start the game
init MMAS
launch 1 iteration (MMAS)
while game not finished do
    wait for the user to position a job (block)
    i ← positioned_job
    j ← position_index
    A_{i,j} ← A_{i,j} × 3
    launch 1 iteration (MMAS)
end
return best found solution
```

Algorithm 1: Algorithm of the MMAS-IEC apporach.

4 Experiments and Results

The experiment aims to show that incorporating human-in-the-loop can positively affect the search process of metaheuristic algorithms, which means better solutions can be found in fewer iterations. Thus the analysis is mainly oriented on comparing MMAS without human-in-the-loop results with MMAS-IEC.

The operation times of the jobs are randomly generated on each new instance of the game since standard benchmark datasets are too big to be visualized in the game. Problem sizes used in the experiment are shown in Table 1. An example of the video game can be seen in Figure 1. The comparison of brute-force algorithm, MMAS algorithm without human-in-the-loop, and MMAS-IEC approach is performed. The results are also compared to the one variant of the Genetic Algorithm (GA) [14].

Table 1: Sizes of the problem in the experiment.

Number of jobs	Number of machines
5	3
10	2
10	4
12	2
12	4
12	6

In both executions of the MMAS algorithm, with and without human-in-the-loop, the parameters were set as follows: $n = 5$ (number of ants), $p_0 = 0.9$ (probability to select the job with highest pheromone), $mmr = 5$ (min-max ratio), $\rho = 0.75$ (persistence of the trail), $max\_iter = 100$ (maximum number of iterations) and $max\_stag = 5$ (maximum number of iterations of stagnation - iterations with the same solution). The parameters of MMAS were set according to the literature [5]. Readers are advised to read the referenced paper for an

[1]Source code available at: https://github.com/Vid201/flow-shop-scheduling-iec

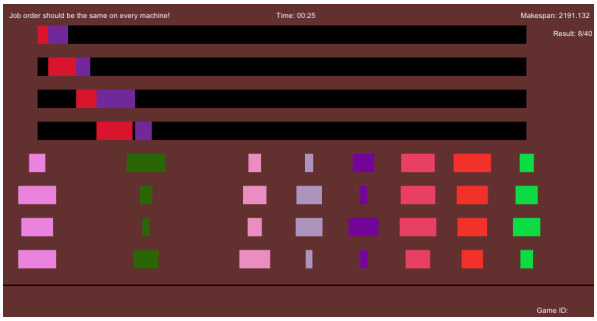

Figure 1: The gameplay of the video game.

in-depth description of the MMAS algorithm. For GA, the parameters were adapted from the used implementation [14].

The MMAS algorithm without human-in-the-loop was run 30 times to get solutions characterized by the algorithm and not randomness. In the MMAS-IEC approach, the multiplier of the pheromone was set to 3, which means element $\mathbf{A}_{i,j}$ is increased if the user places the job i on position j in the game. The size of the multiplier was chosen empirically.

To measure the performance of algorithms, the suboptimal solutions of metaheuristic algorithms were compared to optimal solutions by calculating relative percentage deviation (RPD), which tells how much suboptimal solutions are worse than the optimal solution. RPD was calculated according to the following equation:

$$RPD = \frac{Suboptimal_{sol} - Optimal_{sol}}{Optimal_{sol}} \times 100 \qquad (2)$$

The number of iterations to find the best solution was also compared since time execution was much longer in the interactive approach due to playing the game.

4.1 Analysis and Discussion

The video game and other algorithms were launched five times for each problem size, and the average/mean makespan with standard deviation was calculated. The brute force algorithm was run only on smaller problem sizes since its execution time was too long for the other instances. Where brute force results were available, RPD was also calculated. Results for the experiment are shown in Table 2.

While in the smallest instances of the problem, the MMAS algorithm outperformed the MMAS-IEC approach, the latter achieved better results in all the other problem sizes. This implies that when the search space becomes enormous and complex, human intuition positively affects the algorithm and leads to a better solution. Since the user affected the pheromone level and directed the algorithm towards better solutions, the average number of iterations was less than with only the MMAS algorithm. It must be noted that the best solution in the MMAS-IEC is usually not the same as the end state of the blocks in the user's game. The user only affected the pheromone level of the job on the selected position, and the game's order does not directly change the solution. The best solution is still the solution found by the MMAS algorithm, while the user only changed the algorithm's internal state, i.e., pheromone matrix. Figure 2 shows that when the problem sizes increase, MMAS-IEC tends to find solutions with a lower makespan.

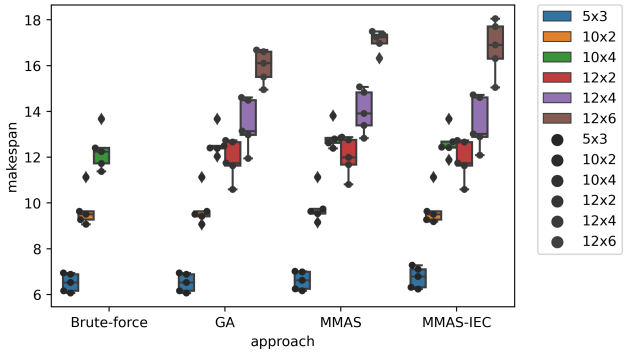

Figure 2: Box plots of makespan for different sizes of the problem with all four approaches. Colors represent different sizes of the problem, while dots show makespans for all five settings per each problem size for each approach.

Results of the MMAS-IEC approach are very dependent on the way the user plays the game. In the first played games, the results of the MMAS-IEC were not better

Table 2: Results of the experiment, where N is number of jobs, M is number of machines, $\overline{M}$ is average makespan, σ is standard deviation, $\overline{IT}$ is average number of iterations for best solution and $\overline{RPD}$ is average relative percentage deviation.

Problem sizes		Approach													
N	M	Brute force		MMAS				GA				MMAS-IEC			
		$\overline{M}$	σ	$\overline{M}$	σ	$\overline{IT}$	$\overline{RPD}$	$\overline{M}$	σ	$\overline{IT}$	$\overline{RPD}$	$\overline{M}$	σ	$\overline{IT}$	$\overline{RPD}$
5	3	6.514	0.402	6.605	0.399	31.0	1.396	6.515	0.402	25.8	0.015	6.742	0.463	3.6	3.500
10	2	9.719	0.814	9.834	0.753	27.8	1.183	9.749	0.797	45.0	0.309	9.742	0.792	2.8	0.236
10	4	12.281	0.876	12.890	0.543	28.4	4.958	12.594	0.625	61.2	2.549	12.614	0.657	5.2	2.711
12	2	/	/	12.019	0.845	26.0	/	11.870	0.877	42.4	/	11.869	0.877	2.6	/
12	4	/	/	14.005	0.949	29.8	/	13.430	1.118	49.2	/	13.463	1.152	5.4	/
12	6	/	/	17.075	0.464	19.8	/	15.970	0.739	84.2	/	16.801	1.195	8.2	/

or were even worse than MMAS alone. But after some games, strategies for the game come into mind, and results get better, which means the makespan decreases. One of the strategies that tend to work well is to position the job with increasing operations times by the machine number in the first place and then putting jobs with as little space between operations as possible on all machines. While the number of iterations is much less in the MMAS-IEC approach, time to find the best solution depends on how fast the user plays the game, e.g., MMAC alone can find the suboptimal solution in 1 second, while the user cannot play the game so fast. For bigger instances, around 10 seconds are needed to find the best suboptimal solution (the game does not have to be played to the end to find the best solution).

Comparing to the GA, both MMAS and MMAS-IEC approaches achieve worse results, except when $N = 10$ and $M = 2$, where the MMAS-IEC approach achieves the best ones. According to the obtained results, combining an interactive process, i.e., gamification, and GA, looks like a promising direction to explore.

The open problem for this gamification process is how to scale the game to bigger problem sizes since the game can become too complex, and it is also hard to visualize blocks and boards. The experiments were successfully conducted on the problems with up to 6 machines and 12 jobs, but scaling the game to more machines and jobs is open for further research.

5 Conclusion

This work shows how human intuition for problem-solving can be incorporated into metaheuristics algorithms for NP-hard problem PFSP. MMAS-IEC approach tends to find better solutions than fully autonomous metaheuristic algorithm MMAS, especially in bigger settings of the problem. Solutions are also found in a fewer number of iterations. But still, there are some open questions to the proposed approach, as bigger instances are hard to visualize, and playing the game usually takes more time than the MMAS algorithm alone. Both of these drawbacks are good starting points for further research.

Further research could be dedicated to selecting a metaheuristic algorithm since this study only accounts for the MMAS algorithm for PSFP. At the same time, several other algorithms proved to be suitable for this problem, e.g., the GA from the analysis. The research goal was to use the same metaheuristic algorithm with and without human interaction and not to compare different metaheuristic algorithms. On the MMAS algorithm itself, other techniques to change the pheromone level could be explored. Various strategies to encourage the player to perform well could also be introduced, e.g., online scoreboards or rewards, which could increase the player's performance with a positive loop. It would also be interesting to compare and try the MMAS-IEC approach on one of the benchmarks for PSFP, e.g., Taillard's benchmark problems [15].

References

[1] D. E. Knuth. Postscript about np-hard problems. *SIGACT News*, 6(2):15–16, Apr. 1974. ISSN: 0163-5700. DOI: 10.1145/1008304.1008305.

[2] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954.

[3] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

[4] M. Nawaz, E. E. Enscore Jr, and I. Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

[5] T. Stützle et al. An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, volume 3, pages 1560–1564, 1998.

[6] A. Holzinger, M. Plass, M. Kickmeier-Rust, K. Holzinger, G. C. Crişan, C.-M. Pintea, and V. Palade. Interactive machine learning: experimental evidence for the human in the algorithmic loop. *Applied Intelligence*, 49(7):2401–2414, 2019.

[7] E. Ignall and L. Schrage. Application of the branch and bound technique to some flow-shop scheduling problems. *Operations research*, 13(3):400–412, 1965.

[8] S. Suliman. A two-phase heuristic approach to the permutation flow-shop scheduling problem. *International Journal of production economics*, 64(1-3):143–152, 2000.

[9] I. Osman and C. Potts. Simulated annealing for permutation flow-shop scheduling. *Omega*, 17(6):551–557, 1989. ISSN: 0305-0483.

[10] E. Vallada, R. Ruiz, and J. M. Framinan. New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3):666–677, 2015. ISSN: 0377-2217.

[11] A. Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016.

[12] N. D. Ross, M. B. Johns, E. C. Keedwell, and D. A. Savic. Human-evolutionary problem solving through gamification of a bin-packing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1465–1473, 2019.

[13] M. Vargas-Santiago, R. Monroy, J. E. Ramirez-Marquez, C. Zhang, D. A. Leon-Velasco, and H. Zhu. Complementing solutions to optimization problems via crowdsourcing on video game plays. *Applied Sciences*, 10(23):8410, 2020.

[14] Suyunu. Github - suyunu/flow-shop-scheduling: genetic algorithm for flow shop scheduling. URL: https://github.com/suyunu/Flow-Shop-Scheduling (visited on 07/23/2021).

[15] E. Taillard. Benchmarks for basic scheduling problems. *european journal of operational research*, 64(2):278–285, 1993.