**Aurelia PĂTRAȘCU, PhD (corresponding author)**
apatrascu@upg-ploiesti.ro
Faculty of Economical Sciences, Petroleum -Gas University of Ploiești, Romania

**Florentina Alina TOADER, PhD**
ftoader@upg-ploiesti.ro
Faculty of Economical Sciences, Petroleum-Gas University of Ploiești, Romania

**Aniela BĂLĂCESCU, PhD**
anielabalacescu@gmail.com
Faculty of Economics "Constantin Brâncuși" University of Târgu Jiu, Romania

# An Improved Multi-Objective Hybrid Algorithm for Solving Job Shop Scheduling Problem

**Abstract.** *The Job Shop Scheduling Problem (JSSP) continues to represent a challenge for researchers from all over the world who are trying to find optimal solutions. The current trend is represented by combining several algorithms in hybrid methods that offer an increased quality of results. This paper proposes a hybrid artificial-based algorithm that aims to minimise the total work flow time, which makes the problem more difficult mathematically speaking but more interesting from a practical point of view. This algorithm exploits a multiple-refined random generation by using an adapted genetic algorithm that becomes the start generation of the hybrid algorithm. In addition, the proposed hybrid algorithm aims to combine particle swarm optimisation and simulated annealing advantages. A complex heuristic function is implemented in order to evaluate each individual solution as accurately as possible. A substantial experimental study across several classic benchmarks was performed in order to demonstrate the algorithm performance; the results are cross compared with other algorithms, and conclusions were drawn. The experimental study confirms the performance and effectiveness of the proposed algorithm, thus providing a significant contribution to the field of JSSP optimisation.*

**Keywords**: *artificial intelligence, evolutionary computation, Job Shop Scheduling.*

**JEL Classification:** C021, C63, C88, O14, O21.

## 1. Introduction

Developing efficient manufacturing systems, adapted to the dynamic challenges of the global economy, requires new technologies and automated software tools to optimise production planning.

The early studies on machine scheduling problems included programming in two and three stages with the inclusion of setup times. These later evolved into flow-shop scheduling problems on two and three machines.

DOI: 10.24818/18423264/58.3.24.11

The JSSP objective is to determine the best order of tasks to be performed on different machines to minimise the makespan (the total time it takes to complete all tasks).

In theory, it is possible to optimally solve the JSSP, but in computational practice it is nearly impossible since it is part of the NP-hard problems (Fakoor and Beheshti, 2021; Garey et al., 1976). Only for small dimension problems (where the machines number and the jobs number does not exceed 10) the problem can be solved by using classical algorithms, but not with an optimal solution (Babu and Jayaraman, 2021).

Over the past six decades, a plethora of works have been published developing various models, approaches, and algorithms for solving machine scheduling problems, including JSSPs (Job Shop Scheduling Problems). These research works have contributed to the development and refinement of methodologies used to tackle these complex problems, providing a solid foundation for optimising production planning in manufacturing systems (Abdullah and Abdolrazzagh-Nezhad, 2014; Gao et al., 2019; Xie et al., 2019).

In the literature, these articles are classified into three main types over a period of time and can provide a useful framework for navigating and understanding the literature on task scheduling programming.

Type A: These articles represent a landmark in understanding and comprehensively analysing multiple classes of task scheduling problems. They go beyond merely presenting the problems and delve deeply into the models, approaches, and algorithms relevant to each problem class. For example, single-machine scheduling involves optimising a single machine to complete a series of tasks efficiently, while parallel-machine scheduling focuses on synchronising multiple machines to complete a common set of tasks.

Type B: This type of article focusses on providing a comprehensive survey for a single type of Job Shop Scheduling Problem (JSSP). They thoroughly analyse the specific characteristics of this particular type of problem and explore methods and techniques that are effective in solving it. For example, a JSSP might involve specific constraints related to the order in which certain tasks must be performed or the time required for each task. These works are valuable for researchers and practitioners who want to focus on a detailed understanding of a particular type of task scheduling problem.

Type C: Work in this category provides specialised and detailed analysis of the approaches and algorithms used in solving shop scheduling problems. They focus exclusively on this subcategory of task scheduling and explore different strategies for optimising processes in a production environment. These articles may examine, for example, specific algorithms for task scheduling in a manufacturing setting or innovative approaches for efficient resource and time management in a shop.

From Figure 1, precise details about the number of works in each type and the relevant publication years for each Type A, Type B, and Type C are depicted. They are simply marked as A, B, and C, respectively. B.1 to B.5 denote the subdivisions

of B, corresponding to dynamic JSSP, JSSP with setup and cost times, flexible JSSP, common due date JSSP, and fuzzy JSSP, respectively. C.1 to C.3 denote the subdivisions of C, where C.1 reviews both exact and approximate algorithms, C.2 refers only to methods with priority rules or dispatching rules, and C.3 is dedicated to reviewing collective intelligence and evolutionary algorithms.

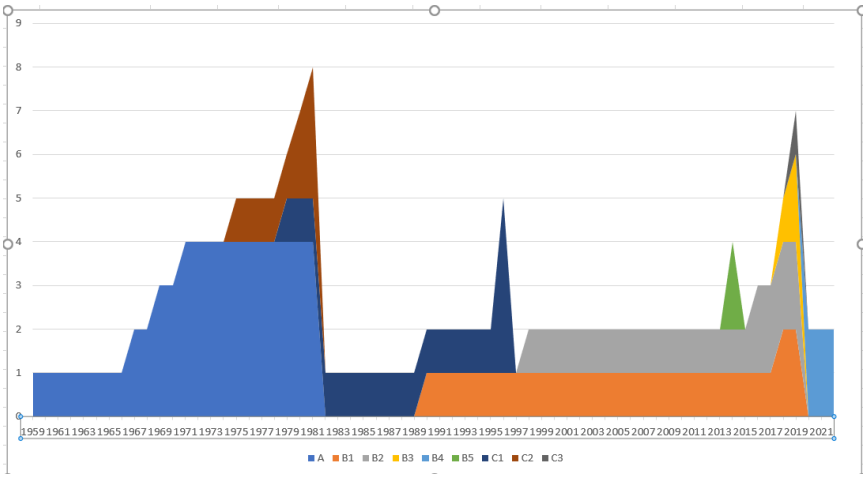


**Figure 1. Multi-agent game model logical relation**
*Source*: Authors' adaptation according to Xiong et al., 2022.

Together, these categories of articles provide a diverse and comprehensive range of information for those interested in the field of task scheduling. They cover both theoretical and practical aspects of solving task scheduling problems and represent a valuable resource for researchers, professionals, and students in this field.

For the most part, the literature focusses on a specific type of problem or a particular issue related to task-scheduling programming, except for some older works from before the 1980s. With the continuous evolution of technology, an increasing number of efficient algorithms have been proposed to solve task scheduling problems.

Comparatively, there are far fewer types of task scheduling problem models than the variety of algorithms used to solve them. Therefore, it is crucial to analyse and summarise the types and characteristics of task scheduling problems from a more detailed perspective to better understand the state of research in this continuously evolving field.

The algorithm proposed in this article is based on a refined multi-stage random generation, utilising an adapted genetic algorithm to create an initial generation of the hybrid algorithm. The proposed algorithm combines the advantages of two distinct optimisation techniques: particle swarm optimisation and simulated annealing. Particle swarm optimisation is efficient in quickly exploring the search space, while simulated annealing can help avoid stagnation in local optima by accepting less favourable solutions with a certain probability.

The research conducted in this paper falls, according to Figure 1, between flexible JSSP (B2) and evolutionary algorithms (C3). By combining these techniques into a hybrid algorithm, the aim is to achieve superior quality solutions in the shortest possible time. This approach can be particularly useful in the context of dynamic and complex production environments, where efficiency in managing resources and production time can make the difference between success and failure.

The current trend is represented by combining several algorithms in hybrid methods that offers an increased quality of results.

The classic scheduling problem JSSP involves scheduling a set of jobs on a set of machines with the goal of minimising the makespan, which is the total time it takes to complete all jobs. The basic characteristics that define this problem are (Pinedo, 2002):

- Jobs and Tasks: Each job is composed of a set of tasks that need to be performed in a specific order. The tasks of a job depend on each other and need to be completed in a specific sequence.
- Machines: Each task needs to be performed on a specific machine. There are a limited number of available machines, and each machine can perform only one task at a time.
- Processing Time: The processing time for each task on each machine is known, and it is constant.
- Constraints: There may be constraints between tasks, such as setup times, order, or machine availability, which must be taken into account when scheduling the jobs.
- Optimisation Objective: The objective of a JSSP is to find the schedule that minimises the makespan, while satisfying all constraints.

The general outline of the mathematical model for the JSSP described in Toader, 2015 consist in the following notations: $M = \{M_1, M_2, \dots, M_m\}$ - the machines set, where m represents the number of machines, $P = \{P_1, P_2, \dots, P_n\}$ a set of products and $S = \{S_1, S_2, \dots, S_n\}$ a set of series for each product, where n represents the number of products. For each $p_i \in P$ an ordered set of corresponding machines are defined, each one with the corresponding amount of time for the specific operation $t_i$.

The solution of the problem is represented in this case by a scheduling operation plan $\pi = (P, S, \pi_i)$, where $\pi_{ij} = (O_i, M_i, A_i, E_i)$, $O_i = \{o_1^i, o_2^i, \dots, o_n^i\}$ represents the ordered set of operations defined for each product $P_i$, $M_i = \{m_1^i, m_2^i, \dots, m_{n_i}^i\}$ represents the ordered set of machines that needs to be accessed in order to fulfill the operations set $O_i$ defined for each product $P_i$, $A_i = \{ta_1^i, ta_2^i, \dots, ta_n^i\}\}$ represents the accessing time list on each machine, $E_i = \{te_1^i, te_2^i, \dots, te_n^i\}$ represents the completion time list for each job on each machine.

The problem constraints are the following:

- Precedence constraints: tasks are performed in the correct order, one machine cannot substitute another.

- Resource constraints: a machine can only perform one task at a time, a single product can be allocated to a given machine at a moment of time.

## 2. Multi-objective hybrid algorithm method

Hybrid artificial intelligence algorithms refer to a combination of two or more AI techniques, such as machine learning, heuristics, and metaheuristics, to solve a scheduling problem. In the context of the Job Shop Scheduling Problem (JSSP), hybrid artificial intelligence algorithms can be applied to improve the efficiency and quality of solutions. Here are some examples of hybrid AI algorithms used in JSSP:

- Genetic Algorithm and Simulated Annealing: A genetic algorithm can be combined with simulated annealing to produce a hybrid algorithm that benefits from the global exploration of the genetic algorithm and the local refinement of simulated annealing. Hybrid algorithms are proposed in Chen et al., 2021; Fahad el al., 2021;
- Particle Swarm Optimisation and Tabu Search: A hybrid algorithm that combines Particle Swarm Optimisation (PSO) and Tabu Search can be used to solve the JSSP. The PSO algorithm provides a global search capability, while the tabu search algorithm performs local refinement to avoid getting stuck in local optima. Hybrid algorithms are proposed in Kuo et al., 2020; Zhang et al., 2022;
- Artificial Neural Networks and Ant Colony Optimization: An Artificial Neural Network can be trained to approximate the processing time of tasks on different machines, and then combined with Ant Colony Optimisation to find an optimal scheduling solution for the JSSP. Hybrid algorithms are proposed in Al-Zoubi et al., 2021; Ma et al., 2022.

Projects that led to the development of some software libraries for scheduling problems are presented in Bräsel, 2010 and Lekin Project, 2023 and share a set of classical or AI based algorithms that can be tested in several situations.

These hybrid AI algorithms can improve the efficiency and quality of solutions compared to using a single AI technique. Ontologies in artificial intelligence applied to JSSP (Job Shop Scheduling Problem) provide a structured framework that enhances AI's capability to model, understand, and solve complex scheduling tasks (Bodea et al., 2010).

The hybrid artificial-based algorithm presented in this paper combines the advantages of three well-known artificial intelligence algorithms: Genetic algorithm (Deb, 2001; Goldberg, 1989), Particle Swarm Optimisation (Coello and Liang, 2020; Elsawah et al., 2016; Li et al., 2021) and Simulated Annealing (Green and Appel, 1981; Marinakis and Marinaki, 2009) and represents the continuation of the research presented in Toader, 2015 and tries to combine the advantages of each of the algorithms, while minimising their disadvantages. Therefore, the goal is to obtain a reliable and efficient algorithm for solving the proposed problem.

This hybrid algorithm exploits a random multiple-refined generation using an adapted genetic algorithm that becomes the start generation of the hybrid algorithm. Further the proposed hybrid algorithm aims to combine the particle swarm optimisation and simulated annealing advantages.

- ***HAIBA Input Parameters***

The input data set for Hybrid Algorithm for Job-Shop Scheduling Problem (HAIBA) is represented by: m– number of available machines, n – number of products that can be processed, $M = \{M_1, M_2, \dots, M_n\}$ – machines list, $P = \{P_1, P_2, \dots, P_n\}$ – products list, $s_i \in S$ – number of batches for each product $P_i, \forall P_i \in P$, $O_i = \{o_1^i, o_2^i, \dots, o_n^i\}$ - the ordered set of specific operations that must be executed in order to obtain the finite product $P_i$, $M_i = \{m_1^i, m_2^i, \dots, m_{n_i}^i\}$- the ordered list of machine allocated for each operation, , $TP_i = \{tp_1^i, tp_2^i, \dots, tp_{n_i}^i\}$ - the ordered list of execution times for each operation on each machine, Nind – number of initial population individuals, Rr – reproduction rate, Mr – mutation rate, SI – number of selected individuals to be included into the particle population, Np – number of particles, $\theta_1 \in (0,1)$ – cognitive parameter, $\theta_2 \in (0,1)$– social parameter, Imax – maximum number of iteration for PSO algorithm, T0 – initial temperature for SA algorithm, Tmin – initial temperature for SA algorithm, α – cooling rate for SA algorithm.

- ***HAIBA Encoding Scheme***

The solution encoding scheme is designed according to the mathematical model presented in Section 2. Each candidate solution is structured as a production plan $\pi = (P_i, s_j, \pi_{ij})$, where $\pi_{ij} = (O_i, M_i, A_{ij})$ and $A_{ij} = \{a_1^{ij}, a_2^{ij}, \dots, a_{n_{ij}}^{ij}\}$ represents a list of accessing time for each job on each machine.

The main objective is to find a plan π that satisfies all constraints, minimises the makespan ($C_{max}$) and ensures that machines distribution is done in the optimal way, minimising as much as possible the idle times ($I_{di}$), the starting time values of each job ($t_{ij}$), and minimising semi-finished products local warehouses overloading (LWL). The objective function fit that evaluates each solution quality is represented in Equation (1).

$$fit = \frac{1}{C_{max} + \sum_{i=1}^{n} Id_i + \sum_{i=1}^{n} \sum_{j=1}^{S_i} t_{ij} + LWL} \quad (1)$$

The solution returned by the hybrid algorithm consists in a production schedule with the best value of the objective function fit. Since all the values considered need to be minimised, the value of the fit function needs to be maximised.

In essence, the hybrid algorithm works to optimise the production schedule in such a way that it achieves the best possible match with the defined objective, helping to improve its efficiency and performance.

- ***HAIBA Description***

The HAIBA approach includes two important stages:

- The initial particle population is randomly generated. But after this step, the GA algorithm will be repeatedly used to refine the results, and the best individuals will be selected to be included in the initial population of particles. This step is repeated until the population of particles is complete.
- The PSO and SA algorithms structures are merged helping each other to move towards the optimal solution.

The detailed schematic representation of HAIBA, using the encoding scheme presented in Section 3.2 is available in Figure 2.

The hybrid algorithm HAIBA combines the advantages of the three methods that are its basis, preventing in this way the blocking of solutions in the area of the local maximum and allowing it to move towards the global maximum.

Although traditionally the PSO algorithm starts from a randomly generated population of particles, this first step of the proposed approach is different: several sets of particles are randomly generated, and refined by means of the AG algorithm. Only the best particles are included in the initial PSO population.

The next step of the algorithm combines PSO and SA as follows:

- Until the maximum number of accepted iterations is reached, the PSO algorithm is applied to the current population, updating the value of Gbest at each step if necessary;
- After updating the positions of the particles, a part of them will be randomly selected and will be subjected to the SA process again. This operation aims to avoid blocking in local maximum values of the candidate solutions;
- With each iteration the particles representing candidate solutions are evaluated using the fitness function presented in equation (1), taking into account the following criteria which need to be minimised to obtain an optimal solution: makespan minimisation (Cmax) idle times minimisation (Idi) and overloading the semi-finished products local warehouses (LWL).

**Experimental Results**

The performance of the HAIBA algorithm is tested with 20 sets of input parameters, as presented in Table 1.

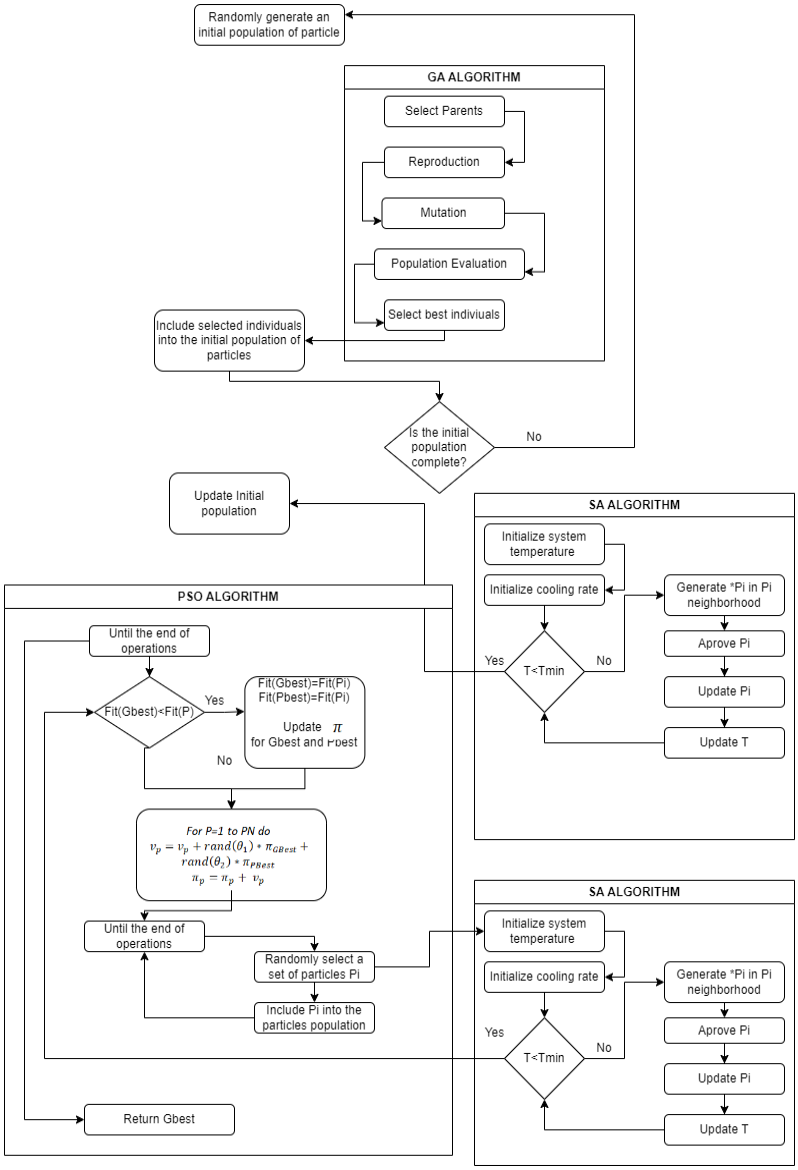


**Figure 2. HAIBA Schematic Representation**
*Source*: Authors' own creation.

The algorithm is implemented in C++ language and tested in various situations, and the 20 input data sets contain the most diverse values of the hybrid algorithm parameters to test the algorithm in the most varied situations. Each input data set consists of values for: Nind – number of individuals, Np – number of particles, ∅1 – cognitive parameter, ∅2 – social parameter, Imax1- maximum number of iteration for the genetic algorithm section of HAIBA, Imax2 – maximum number of iteration for the genetic algorithm section of HAIBA, T0 –

initial temperature for the simulated annealing section of HAIBA, Tmax – maximum accepted temperature for the simulated annealing section of, α – cooling rate for the simulated annealing section of HAIBA, Rr – reproduction rate for the genetic algorithm section of HAIBA and Mr - mutation rate for the genetic algorithm section of HAIBA.

In addition to the values completed in Table 1, the maximum accepted temperature parameter has two fixed values: 0.01 for PSO01-PSO10 parameter sets and 0.005 for PSO11-PSO20 parameter sets.

**Table 1. Input Parameter Set**

| Parameter Set | Nind | Np | $\emptyset_1$ | $\emptyset_2$ | $I_{max1}$ | $I_{max2}$ | T0 | α | Rr | Mr |
|---|---|---|---|---|---|---|---|---|---|---|
| **PS01** | 50 | 100 | 0.2 | 0.2 | 50 | 50 | 25 | 0.01 | 0.4 | 0.01 |
| **PS02** | 50 | 100 | 0.3 | 0.2 | 80 | 50 | 25 | 0.01 | 0.5 | 0.01 |
| **PS03** | 50 | 100 | 0.4 | 0.2 | 50 | 80 | 25 | 0.01 | 0.6 | 0.01 |
| **PS04** | 50 | 100 | 0.4 | 0.3 | 50 | 50 | 25 | 0.01 | 0.3 | 0.01 |
| **PS05** | 50 | 100 | 0.4 | 0.4 | 70 | 70 | 20 | 0.01 | 0.2 | 0.01 |
| **PS06** | 75 | 100 | 0.6 | 0.2 | 80 | 70 | 20 | 0.01 | 0.4 | 0.005 |
| **PS07** | 75 | 100 | 0.2 | 0.6 | 50 | 50 | 20 | 0.005 | 0.5 | 0.005 |
| **PS08** | 50 | 100 | 0.4 | 0.6 | 80 | 50 | 25 | 0.005 | 0.6 | 0.005 |
| **PS09** | 50 | 100 | 0.6 | 0.4 | 50 | 80 | 25 | 0.005 | 0.3 | 0.005 |
| **PS10** | 50 | 100 | 0.8 | 0.4 | 50 | 50 | 25 | 0.005 | 0.7 | 0.005 |
| **PS11** | 75 | 150 | 0.2 | 0.2 | 70 | 70 | 25 | 0.005 | 0.4 | 0.01 |
| **PS12** | 75 | 150 | 0.3 | 0.2 | 80 | 70 | 20 | 0.01 | 0.5 | 0.01 |
| **PS13** | 75 | 150 | 0.4 | 0.2 | 50 | 50 | 20 | 0.01 | 0.6 | 0.01 |
| **PS14** | 100 | 150 | 0.4 | 0.3 | 80 | 50 | 20 | 0.01 | 0.3 | 0.01 |
| **PS15** | 100 | 150 | 0.4 | 0.4 | 50 | 80 | 30 | 0.01 | 0.7 | 0.007 |
| **PS16** | 100 | 150 | 0.6 | 0.2 | 50 | 50 | 30 | 0.01 | 0.4 | 0.007 |
| **PS17** | 100 | 150 | 0.2 | 0.6 | 70 | 70 | 30 | 0.005 | 0.5 | 0.007 |
| **PS18** | 75 | 150 | 0.4 | 0.6 | 80 | 70 | 25 | 0.005 | 0.6 | 0.007 |
| **PS19** | 75 | 150 | 0.6 | 0.4 | 50 | 50 | 25 | 0.005 | 0.3 | 0.01 |
| **PS20** | 75 | 150 | 0.8 | 0.4 | 70 | 50 | 25 | 0.005 | 0.7 | 0.01 |

*Source*: Authors' own creation.

In order to validate the algorithms' performances, a set of 25 classical benchmarks that are well known in the scientific literature were used (Table 2), he results are compared with the optimal value and with the accepted range consisting in the lower and upper bound (Talliard, 1993; Benchmarks for basic scheduling problems, 2023; CSP2SAT: JSS benchmark results, 2023).

Also, Table 2 contains the results obtained for the same benchmarks test by using the classical SA and PSO algorithms, and also a hybrid algorithm detailed in paper Toader, 2015. The table 2 structure is as follows: BM –the benchmark name as known in the scientific literature, Dimension (Machines x Jobs) – machines

number and job number, LB (Lower Bound) and UB (Upper Bound) - the range of values accepted for the specific benchmark in the specialized literature, OV (Optimal Value) – the optimal value accepted for the specific benchmark in the specialised literature , PSO Optimal – best value obtained by using PSO, SA Optimal – best value obtained by using SA, H-PSO-SA – best value obtained by using H-PSO-SA, HAIBA Optimal – best value obtained by using HAIBA.

**Table 2. Experimental Results**

| BM | DMJ | LB | UB | OV | PSO Optimal | SA Optimal | H-PSO-SA Optimal | HAIBA Optimal |
|---|---|---|---|---|---|---|---|---|
| **M06** | 6 x 6 | 46 | 68 | 55 | 72 | 70 | 55 | 55 |
| **M10** | 10 x 10 | 742 | 1071 | 930 | 1130 | 1139 | 930 | 930 |
| **ABZ5** | 10 x 10 | 868 | 1370 | 1234 | 1500 | 1480 | 1234 | 1240 |
| **ABZ6** | 10 x 10 | 742 | 1071 | 943 | 1041 | 1158 | 948 | 946 |
| **LA01** | 10 x 5 | 666 | 830 | 666 | 900 | 932 | 671 | 669 |
| **LA02** | 10 x 5 | 635 | 744 | 655 | 785 | 1003 | 655 | 655 |
| **LA03** | 10 x 5 | 588 | 773 | 597 | 921 | 989 | 598 | 597 |
| **LA04** | 10 x 5 | 537 | 839 | 590 | 1005 | 1107 | 590 | 590 |
| **LA05** | 10 x 5 | 593 | 677 | 593 | 908 | 984 | 789 | 593 |
| **LA15** | 20 x 5 | 1207 | 1442 | 1207 | 1583 | 1622 | 1317 | 1301 |
| **LA16** | 10 x 10 | 717 | 1230 | 945 | 1426 | 1499 | 1280 | 945 |
| **LA17** | 10 x 10 | 683 | 894 | 784 | 1003 | 1089 | 920 | 788 |
| **LA18** | 10 x 10 | 663 | 1128 | 848 | 1337 | 1365 | 934 | 848 |
| **LA19** | 10 x 10 | 685 | 1046 | 842 | 1225 | 1356 | 842 | 842 |
| **LA20** | 10 x 10 | 756 | 1210 | 902 | 1290 | 1411 | 902 | 902 |
| **ORB1** | 10 x 10 | 695 | 1456 | 1059 | 1652 | 1686 | 1059 | 1063 |
| **ORB2** | 10 x 10 | 671 | 1157 | 888 | 1202 | 1214 | 888 | 888 |
| **ORB3** | 10 x 10 | 648 | 1297 | 1005 | 1394 | 1399 | 1005 | 1005 |
| **ORB4** | 10 x 10 | 759 | 1356 | 1005 | 1399 | 1508 | 1020 | 1005 |
| **ORB5** | 10 x 10 | 630 | 1116 | 887 | 1229 | 1381 | 887 | 891 |
| **ORB6** | 10 x 10 | 715 | 1342 | 1010 | 1492 | 1569 | 1280 | 1010 |
| **ORB7** | 10 x 10 | 286 | 502 | 397 | 720 | 839 | 550 | 410 |
| **ORB8** | 10 x 10 | 585 | 1139 | 899 | 1329 | 1444 | 1180 | 921 |
| **ORB9** | 10 x 10 | 661 | 1376 | 934 | 1637 | 1756 | 1380 | 938 |
| **ORB10** | 10 x 10 | 681 | 1284 | 944 | 1529 | 1683 | 1302 | 944 |

*Source*: Authors' own creation.

After the tests carried out, the results highlight the fact that the classic AI implemented algorithms (PSO and SA) fail to fit into the interval of the expected values delimited by the Lower Bound and Upper Bound. Therefore, there is a limitation to the effectiveness of these algorithms in solving the optimisation problem.

The same tests highlight the fact that the hybrid algorithm described in Toader, 2015 falls within the accepted range in 72% of cases, and the HAIBA hybrid algorithm described in this paper falls within the range described by the lower bound and the upper bound in 100% of cases. It can be stated that the HAIBA hybrid algorithm can be a preferred choice over other approaches, including the classical PSO and SA algorithms, being able to generate solutions respecting the quality criteria.

Related to reaching the optimal value for each of the 25 considered benchmarks, the H-PSO-SA algorithm manages to reach this value in 44% of the cases, and the HAIBA algorithm manages to reach this value in 64% of the considered cases. Thus, an improvement in the quality of the hybrid algorithm compared to its previous version can be observed and it can be emphasised that the combination of the advantages offered by AG, PSO and SA in a hybrid algorithm represents a suitable direction to solve the proposed problem in an optimal way.

In Figure 3 the HAIBA performances are presented along all the test data and the results are included in Table 2.


**Figure 3. HAIBA Performance**
*Source*: Authors' own creation.

Figure 4 highlights even more clearly the quality of the results obtained by the proposed hybrid algorithm. These results are pointed up in comparison with the two key reference values from the specialised literature (LB - Lower Bound and UP - Upper Bound).

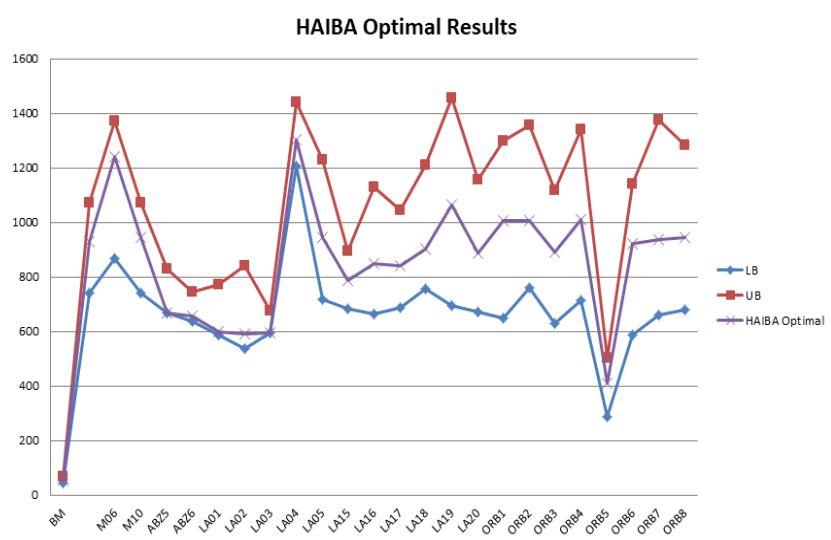


**Figure 4. HAIBA Optimal Result**
*Source*: Authors' own creation.

The analysis of how the results of the proposed hybrid algorithm manage to reach the optimal value accepted for the data sets taken into consideration is presented in Figure 5. It can be seen that the hybrid algorithm manages to reach the optimal value in 72% of the cases, and in the others the value obtained is at the minimum difference compared to this desired optimal value.

It can be concluded that this algorithm manages to handle in situations of flexible lines of different sizes, with a large number of machines and operations. In the case of these extremely complex problems, the situations in which the algorithms make to approach and reach the optimal values are quite limited.

Thus, it can be highlighted even more that the quality of the solutions offered by HAIBA is clearly superior to the quality of the solutions offered both by the classical AI algorithms implemented, and by the other hybrid algorithm that was tested under the same conditions.

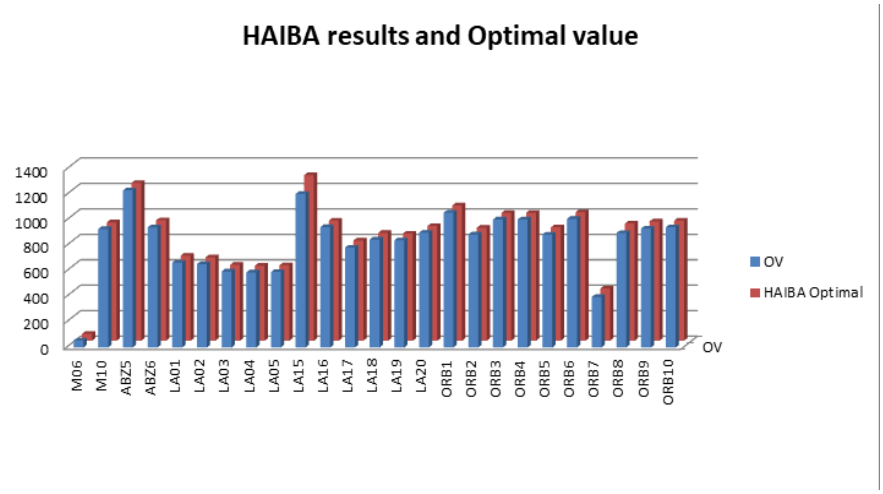


**Figure 5. HAIBA Optimal Results and Optimal Values Comparation**
*Source*: Authors' own creation.

## 3. Conclusions

Taking into account that the JSSP problem is a complex NP-problem, the researchers' attention is directed towards finding solutions to solve it in a manner as close as possible to the optimum. This research proposes a hybrid algorithm that combines the advantages of specific classic AI algorithms (PSO, SA, and AG), thus avoiding the blocking of solutions at the local optima points and thus directing it towards the global optima. The HAIBA algorithm was tested with a set of 25 classic benchmarks from the specialised literature and the solutions quality of the obtained was evaluated using a complex fitness function.

The experimental results highlight the fact that the proposed hybrid algorithm manages to obtain very good results for the considered test data, the solutions fall 100% of the time in the accepted range delimited by the lower bound and upper bound, and in 72% of the cases the optimal accepted value is reached in specialised literature.

These results suggest that the proposed hybrid algorithm is an efficient and reliable approach to solving complex problems and can be successfully used in a variety of fields and practical applications.

The superiority of the results achieved by the proposed hybrid algorithm is further emphasised. These outcomes are juxtaposed with two pivotal benchmarks from the specialised literature, namely the Lower Bound (LB) and Upper Bound (UB).

The obtained results illustrate the fact that the hybrid algorithm aligns with the accepted optimal value for the considered datasets. It is evident that the hybrid algorithm achieves the optimal value in 72% of the instances, while in others the deviation from this optimal value is minimal. Hence, it can be inferred that this algorithm performs admirably in scenarios involving flexible lines of varying sizes,

numerous machines, and operations. Notably, in highly intricate problem scenarios, achieving or closely approaching optimal values remains a challenging task for algorithms.

Future research directions will be oriented towards testing this algorithm on data taken from a real flexible manufacturing line, as well as improving the algorithm by taking into account machine breakdowns/failures (including rescheduling) or compliance with the delivery time of the products. Also, another direction of future research refers to the development of a software solution that contains a library of computerised solvers based on AI and hybrid AI techniques, which may be able to provide solutions for a given JSS problems structure.

***Acknowledgements:*** *This research was funded by a grant of the Petroleum-Gas University of Ploiesti, project number GO-GICS-11063/08.06.2023, within the Internal Grant for Scientific Research.*

## References

[1] Abdullah, S., Abdolrazzagh-Nezhad, M. (2014), *Fuzzy job-shop scheduling problems: A review*. *Information Sciences*, 278, 380-407, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2014.03.060.

[2] Al-Zoubi, S., Abdul-Rahman, S.O., Al-Betar, M. (2021), *An artificial neural network and ant colony optimization hybrid algorithm for job shop scheduling*. *International Journal of Advanced Intelligence Paradigms*, 13(3-4), 262-276.

[3] Babu, A.V.K., Jayaraman, M.A. (2021), *Complexity analysis of job shop scheduling with sequence-dependent setup times*. *Journal of Intelligent Manufacturing*, 32(3), 553-564.

[4] Benchmarks for basic scheduling problems (2023), http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html, accessed on February 2023.

[5] Bodea C.N., Niţchi St., Elmas C., Tănăsescu, A., Dascălu M., Mihăila A. (2010), *Modeling project management competencies using an ontological approach. Economic Computation and Economic Cybernetics Studies and Research*, 44(2), 33-47.

[6] Bräsel, H., Engelhardt, F., Werner, F. (2010), *LiSA - a Library of Scheduling Algorithms: Handbook for Version 3.0*, Univ., Fak. für Mathematik.

[7] Chen, W., Chen, C., Chen, K., (2021), *A Hybrid Genetic Algorithm with Simulated Annealing for Job Shop Scheduling Problems with Non-identical Machines*. *Journal of Industrial and Management Optimization*, 17(2), 1051-1065.

[8] Coello, C.A., Liang, J.J. (2020), *Particle swarm optimization: a survey of historical and recent developments*. *Swarm Intelligence*, 14(1), 1-31.

[9] CSP2SAT: JSS benchmark results (2023), https://cspsat.gitlab.io/csp2sat/jss/, accessed on April 2023.

[10] Deb, K. (2001), *Multi-Objective Optimization using Evolutionary Algorithms*, New York, NY: Wiley.

[11] Dhiflaoui, M., Nouri, H.E., Driss, O.B. (2018), *Dual-Resource Constraints in Classical and Flexible Job Shop Problems: A State-of-the-Art Review*. *Procedia Computer Science*, 126, 1507-1515, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.08.123.

[12] Elsawah, M., Abbass, A., Sarker, R., Cornforth, D.J. (2016), *A review of particle swarm optimization: Part II - hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications*. *Knowledge-Based Systems*, 114, 1-31.

[13] Fahad, A.S.M., Sait, S.M., Ali, A.M.A. (2021), *A Hybrid Genetic Algorithm with Simulated Annealing and Gravitational Search Algorithm for Job Shop Scheduling Problem*. *IEEE Access*, 9, 12730-12744.

[14] Fakoor, M.R., Beheshti, A.H. (2021), *A computational complexity analysis of a job shop scheduling problem with family setups*. *Journal of Industrial Engineering International*, 17(4), 807-818.

[15] Gao,K., Cao, Z., Zhang, L., Chen, Z., Han Y., Pan, Q. (2019), *A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. IEEE/CAA Journal of Automatica Sinica*, 6(4), 904-916.

[16] Garey, M.R., Johnson, D.S., Sethi, R. (1976), *The complexity of flowshop and jobshop scheduling*. *Mathematics of Operations Research*, 1(2), 117-129.

[17] Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Boston, MA: Addison-Wesley.

[18] Green, G.I., Appel, L.B. (1981), *An empirical analysis of job shop dispatch rule selection*. *Journal of Operations Management*, 1(4), 197-203, ISSN 0272-6963, https://doi.org/10.1016/0272-6963(81)90025-5.

[19] Hirsch, M. J. (2017), *Simulated annealing for job shop scheduling with fuzzy processing times*. *International Journal of Production Research*, 55(16), 4819-4834.

[20] Kuo, K.J., Hsieh, C.T., Wang, K.H. (2020), *A hybrid PSO and tabu search algorithm for solving job shop scheduling problem with time lags*. *International Journal of Production Research*, 58(4), 1144-1166.

[21] Lekin Project (2023), https://pypi.org/project/lekin/, accessed on May 2023.

[22] Li, X., Jiang, C., Tang, L. (2021), *A survey of particle swarm optimization and its applications*. *Applied Soft Computing*, 109, Article ID 107022.

[23] Ma, X., Ma, H., Shi, Y. (2022), *A hybrid artificial neural network and ant colony optimization algorithm for job shop scheduling problems with different objectives*. *Soft Computing*, 26(2), 1371-1385.

[24] Marinakis, D., Marinaki, P. (2009), *A survey of simulated annealing applications to operations research problems*. *Journal of the Operational Research Society*, 60(S1), S114-S125.

[25] Pinedo, M. (2002), *Scheduling: Theory, Algorithms, and Systems*, *Springer*.

[26] Talliard, E. (1993), *Benchmarks for basic scheduling problems*. *European Journal of Operational Research*, 64, 278-285.

[27] Toader, F.A. (2015), *A Hybrid Algorithm for Job Shop Scheduling Problem, Studies in Informatics and Control*, 24(2), 171-180, https://doi.org/10.24846/v24i2y201505.

[28] Xie, J., Gao, L., Peng, K., Li, X., Li, H. (2019), *Review on flexible job shop scheduling. IET Collaborative Intelligent Manufacturing*, 1(3), 67-77, https://doi.org/10.1049/iet-cim.2018.0009.

[29] Xiong, H., Shi, S., Ren, D., Hu, J. (2022), *A survey of job shop scheduling problem: The types and models. Computers & Operations Research*, 142, 105731, ISSN 0305-0548, https://doi.org/10.1016/j.cor.2022.105731.

[30] Zhang, Y., Hu, X., Cao, X., Wu, C. (2022), *An efficient hybrid integer and categorical particle swarm optimization algorithm for the multi-mode multi-project inverse scheduling problem in turbine assembly workshop, Computers & Industrial Engineering*, 169, Article ID 108148, https://doi.org/10.1016/j.cie.2022.108148.