An adaptive hybrid approach: Combining genetic algorithm and ant colony optimization for integrated process planning and scheduling

Received 9 February 2018
Revised 22 November 2018
Accepted 9 December 2018

Mehmet Fatih Uslu
Department of Industrial Engineering, Istanbul Rumeli University, Istanbul, Turkey
Süleyman Uslu
Department of Computer and Information Science, Indiana University–Purdue University Indianapolis, Indianapolis, Indiana, USA, and
Faruk Bulut
Department of Computer Engineering, Halic University, Istanbul, Turkey

Abstract

Optimization algorithms can differ in performance for a specific problem. Hybrid approaches, using this difference, might give a higher performance in many cases. This paper presents a hybrid approach of Genetic Algorithm (GA) and Ant Colony Optimization (ACO) specifically for the Integrated Process Planning and Scheduling (IPPS) problems. GA and ACO have given different performances in different cases of IPPS problems. In some cases, GA has outperformed, and so do ACO in other cases. This hybrid method can be constructed as (I) GA to improve ACO results or (II) ACO to improve GA results. Based on the performances of the algorithm pairs on the given problem scale. This proposed hybrid GA-ACO approach (hAG) runs both GA and ACO simultaneously, and the better performing one is selected as the primary algorithm in the hybrid approach. hAG also avoids convergence by resetting parameters which cause algorithms to converge local optimum points. Moreover, the algorithm can obtain more accurate solutions with avoidance strategy. The new hybrid optimization technique (hAG) merges a GA with a local search strategy based on the interior point method. The efficiency of hAG is demonstrated by solving a constrained multi-objective mathematical test-case. The benchmarking results of the experimental studies with AIS (Artificial Immune System), GA, and ACO indicate that the proposed model has outperformed other non-hybrid algorithms in different scenarios.

Keywords Optimization problems, Hybrid heuristics, IPPS, GA, ACO

Paper type Original Article

Applied Computing and Informatics
Vol. 18 No. 1/2, 2022
pp. 101-112
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2210-8327
DOI 10.1016/j.aci.2018.12.002

1. Introduction

Both process planning and scheduling terms have a paramount importance in many industrial processes, especially in manufacturing systems. In this field, the process planning term determines production steps according to the product specifications. Besides, scheduling concept determines how resources can be used according to the process plan. Primitive and traditional optimization methods normally handle this problem in a sequential order. On the other hand, with the increase of the computational capacity of processors, using process planning and scheduling in a hybrid method simultaneously becomes more popular.

Many scheduling problems in the real world applications cannot be accurately solved in polynomial time with any known algorithms. These types of problems are not described in the *P* complexity class. Therefore, metaheuristic algorithms are widely preferred today in real scheduling problems in industries from the 1970s up to now [1]. As one of the most popular optimization methods, Genetic Algorithm (GA) and its hybrid variation are one of the widely used algorithm for integrated planning and scheduling systems [2]. Its mechanism is inspired by natural selection. Ant Colony Optimization (ACO) is another algorithm for solving Integrated Process Planning and Scheduling (IPPS) problems to minimize the maximum completion [3]. This type of optimization method is time-based on crossover and mutation mechanism which is inspired by ants which search food in a graph.

In this field, there are various experimental and theoretical studies in the literature. Some valuable studies are listed in a chronological order below. Firstly, Allahverdi and Aldowaisan presented several new heuristic algorithms for multi-machine non-waiting flow-type problems, taking into account the total completion time, and show that these new approaches are better in terms of error performance than known approaches, including the newly developed GA [4].

Tseng and Lin have presented a hybrid GA to solve the non-wait flow-type scheduling problems with the goal of completion time. This presented algorithm combines a new local search scheme with GA. The local search algorithm combines Insertion Search with Cut-and-Repair algorithms [5].

Li et al. have investigated two different data mining techniques for their study. These techniques are artificial neural networks and binary logistic regression methods. They have evaluated their approach to graphically based hyper-intuitive solutions proposed for test scheduling problems. Time complexity analysis has shown that artificial neural networks and binary logistic regression method accelerate the study. They have assisted in the development of more sophisticated information-based decision support systems [6].

Araujo and Nagano have investigated the scheduling problems in order to minimize the execution time. This problem is famous for being NP-hard, and this problem made a small contribution. In their study, they proposed a new constructive heuristic method named GAP Heuristics with structural property base [7]. The proposed approach is based on two well-known methods in the literature, such as TWOs proposed by Bianco, Dell'Olmo and Giordani [8] and intuitive TRIPS (Triple) [9], which is proposed by Brown, McGarvey, and Ventura are superior in terms of required computational time.

Chaundry and Mahmood have developed an unprecedented flow type scheduling using genetic algorithms. Non-standing flow type scheduling is a limited flow type schedule that is commonly found in manufacturing systems. In this research, it is considered that the total completion time is minimized for N number of jobs processed in M machines using general purpose table based GA. The proposed approach solution is compared with the problems already published in the literature. The proposed approach produces the most appropriate solution for all situations. It also demonstrates that an objective function can be minimized by using the same model without changing the general conception of the GA [10].

Prot et al. model an industrial workshop scheduling problem in their articles as a multi-modal production line type workshop. In the problem they deal with, there are additional constraints such as sequence-dependent preparation times and delivery dates. The problem in decision makers is to minimize the biggest delay. To solve the problem, a taboo search procedure has introduced a valid lower bound to measure this taboo search procedure [11].

Gamma and Singhal have tried to find the ideal table order with GA for flow type scheduling problems involving M machines and N jobs with time-dependent and job partitioned preparations in order. Authors who focus on two types of case studies, both traditional and general, have shown that the optimized time-to-completion value can be accessed with multiple different business sequences instead of one, and can help reduce the time to completion in the scheduling process [12].

Vidal et al. have contributed a component-based heuristic approach to the development of an efficient, applicable and general-purpose algorithm for vehicle route problems and the determination of the challenges in this area. As a result of extensive computational experiments, the method has demonstrated a remarkable performance as well as the most successful problem-oriented algorithms in the literature, or better than them [13].

Pacini et al. have investigated distributed job scheduling efforts for Parameter Scan Experiments (PSE) with bio-inspired techniques in their work. They have created a taxonomy for organizing and analyzing the investigated materials. They point out the strengths and weaknesses of the present experiments. This area describes the work that can be done in the future [14].

Burdett and Kozan have addressed the problem of creating train timings in their work. Train time-table creation is a complicated problem in terms of delays and facilities. They have developed numerically efficient algorithms to define the delay effect in terms of the affected operations. The adjustments and delay values of the affected investigations are spread by the differential graphical model of train surveys. The results of the proposed sensitivity analysis were used to determine program integrity. The analyzes provided information that could be used as part of the proactive scheduling approach. Affected processes can be used to develop meta-intuitive approaches to the chart [15].

Pugazhenthi and Xavior are approaching the primary goal of minimizing the time to complete flow type scheduling problems for M machines with N jobs. In order to solve the flow-type scheduling problem on a modern production framework, EPDT (Extended Prim-Dijkstra Tradeoff) has proposed meta-intuitive approaches called as the BAT (Bat). They applied these two algorithms together with GA for further development in achieving the minimum execution time. In order to measure the performance of these new heuristic approaches, MATLAB solved the problem of benchmarking Taillard of different sizes. GA-applied FPDT heuristic approach for flow-type problems and GA-applied BAT meta-heuristic approach are effective in finding a better set of solutions to solve scheduling problems and to reduce the completion time [16].

Laha and Sapkal propose a heuristic algorithm to minimize the total flow time in the non-waiting flow type scheduling in their work. In experiments, the proposed heuristic approach has outperformed well-known intuition apart from time complexity. Statistical significance tests have proved the superiority of the method [17].

Dey et al. have proposed meta-intuitions to make multilevel thresholding faster in their work. They used quantum mechanics to propose six different quantum-inspired meta-intuitive methods. The results of the six proposed quantum meta-heuristic methods are discussed in order to establish consensus results. Quantum-inspired particle flock optimization is superior to other methods. The computational complexities of the proposed methods are explained in order to find the time-out efficiency of these methods [18].

Kianfara and colleagues have worked on a flexible flow-type system based on the non-deterministic dynamic development for jobs and the sequence-dependent preparation time. The problem is to specify a schedule that minimizes the average delay of the intended tasks. Since the class of the problem is NP-hard, a new shipment rule and a hybrid GA have been developed. The two new methods they have included in their research and the most commonly used shipping conventions in the literature have been combined in the simulation model. The results show that the methods they propose in their work are better than the traditional shipping rules [19].

Li and Gao lately have published a book, summarizing a series of extended researches study on IPPS. They have focused on details of novel solution techniques, discussing the properties, and applications of process planning and scheduling under different environments [20].

Various algorithms like GA, Artificial Immune System, and ACO which aim to solve Scheduling problems are combined and an HTML page & open source Javascript Library is developed as an interface which allows users to compare their algorithms with others in a graphical interface. Users can create various types of Scheduling problems and solve through these algorithms in this application. Also, parameters of GA are optimized for Scheduling problems and via these parameters. Furthermore, a hybrid algorithm is developed using ACO and GA.

In this study, a hybrid approach using GA and ACO called as hAG has both theoretically and empirically presented. The basic approach of the proposed system hAG is to solve one of the two optimization methods first, then to try to improve the solution with the other one. For this reason, the starting algorithm should be chosen wisely at the initial state for better performance. Namely, this approach suggests to run both algorithms simultaneously first, then monitoring their performance to differentiate the better one. This selection criterion makes this proposed approach unique when compared with the others.

This paper has four main sections. In the first section, as placed above there is an introduction to this study with a literature review concerning related studies with this proposed method. In the second chapter below, there are explanations of IPPS problems. In the third section, there are and details of the suggested technique. In the fourth section, experimental studies are presented. In the last section, contributions are summarized.

2. IPPS problem definition

IPPS problems are optimization problems which include both process planning and scheduling. In this study, operations can work on different machines with possibly different running times. This is known as operation flexibility or machine flexibility. There are J jobs and a job consists of P operations which have to be done sequentially. Also, there is M non-identical machine for assigning operations according to their performance. The aim is basically to find minimum makespan [21,22].

The sequence of jobs is alterable but in a specific job, the sequence of operations of a job should be in given order. Any process of any job can operate in any machine which is allowed in the given table.

J	number of jobs
P	number of operations
M	number of machines
i,l	index for jobs, $1 \leq i \leq J$
j,m	index for operations, $1 \leq j \leq P$
k,n	index for machines, $1 \leq k \leq M$
d_{ijk}	1, if jth operation of ith job runs on kth machine. 0, otherwise.
p_{ijk}	processing time of jth operation of ith job on kth machine.
x_{ijk}	completion time of jth operation of ith job on kth machine.

Our main objective is minimizing total makespan.

Minimize $F = {}_{\forall\ \mathrm{i,j,k}} \max \{x_{ijk} * d_{ijk}\}$
subject to

$$x_{ijk} - x_{imn} \geq p_{ijk} \quad \forall i, j, k, m \, and \, n : j > m \tag{1}$$

$$x_{ijk} - x_{lmk} \geq p_{ijk} \, \forall i, j, k \, and \, m\text{: } j^{th} \, process \, of \, i^{th} \, job \, runs \, after \, m^{th} \, process \, of \, l^{th} \, job \, on \, k^{th} \, machine. \tag{2}$$

$$\sum_{k=1}^{M} d_{ikp} = 1 \, \forall i \, \text{and} \, k \tag{3}$$

$$x_{ijk} \geq p_{ijk} \, \forall i, j \, and \, k : i^{th} \, has \, j^{th} \, process \, runs \, on \, k^{th} \, machine. \tag{4}$$

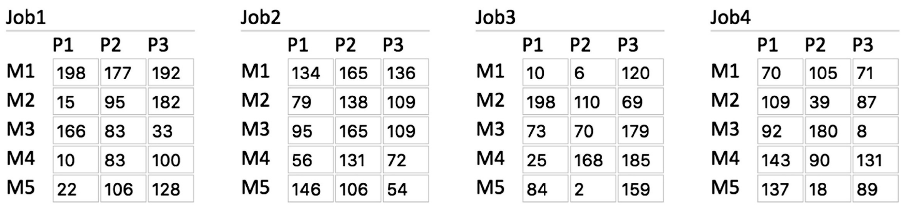
$$d_{ijk} \in \{0, 1\} \, \forall i, j \, and \, k \tag{5}$$

As it is seen there are some related limitations given above about the IPPS problems. The first constraint (1) implies that the tasks of every customer arrange are handled by the priority required. Constraints (2) guarantees that any two tasks having a place with a similar customer arrange cannot be handled in the meantime. Constraint (3) guarantees that just a single resource for every activity ought to be chosen. Finally, constraints (4) and (5) infer non-negativity and integrality of the corresponding variables [23].

Figure 1 shows a random generated IPPS problem variable table. This table shows machines' performances with respect to the product and machine specifications. According to this table, all jobs have three operations which have to be done sequential, and these operations can be done in one of five machines. It means J = 4, P = 3, and M = 5. If we choose Operation 1 of Job 1 performs on Machine 1, it cost us 198-unit time.

Figure 2 shows an example Gantt diagram of a solution to the given scenario. In this solution, three operations run in Machine 1. 2nd operation of Job 4, 2nd operation of Job 2 and 3rd operation of Job 1 are them. Makespan of this solution is 580 because 3rd machine ends last. Figure 2 also shows the priority of jobs. 2nd operation of Job 4 waited for 1st operation of the 4th job on Machine 5. The adaptive GA and ACO parameters for the solution given in Figure 2 is as follows as seen in Table 1.

In GA, there are some mutation types such as Bit Flip mutation, Swap mutation, Scramble mutation, and Inversion mutation. All of these have some specific techniques. Similarly, there are some crossover types such as Single point crossover, Two-point crossover, Uniform



Job1

	P1	P2	P3
M1	198	177	192
M2	15	95	182
M3	166	83	33
M4	10	83	100
M5	22	106	128

Job2

	P1	P2	P3
M1	134	165	136
M2	79	138	109
M3	95	165	109
M4	56	131	72
M5	146	106	54

Job3

	P1	P2	P3
M1	10	6	120
M2	198	110	69
M3	73	70	179
M4	25	168	185
M5	84	2	159

Job4

	P1	P2	P3
M1	70	105	71
M2	109	39	87
M3	92	180	8
M4	143	90	131
M5	137	18	89

Figure 1. An example IPPS problem table.

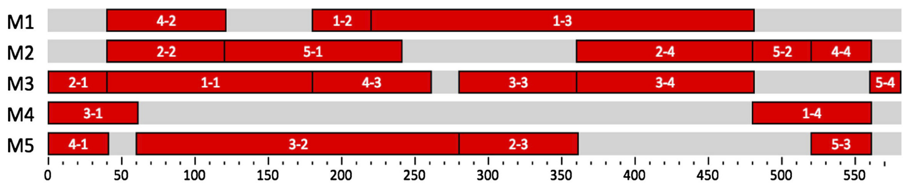


Figure 2. A sample IPPS problem solution.

crossover, and Arithmetic crossover. Generally, the adaptive parameter might be chosen experimentally in order to achieve better performance. In the empirical studies, Swap mutation and as Single point crossover have been chosen as the most adjusting parameters.

As these optimization methods might outperform depending on the best-selected parameters, all these values have been observed in the tests.

3. Proposed hybrid approach (hAG)

It will be better to remember the general aspects of both ACO, GA, and AIS techniques before explaining the proposed method.

3.1 Ant colony optimization (ACO)

This proposed hybrid approach (hAG) uses the ACO algorithm design which proposed in [24]. This ACO design takes IPPS problem as a graph. In that graph, every process on every machine that can execute that process is a node. All nodes that represent a process of the job has a directed arc to every node that represents the next process of that job and undirected arc to any process of other jobs. In addition, a node is located at the start point, has directed arcs to nodes that represent the first processes of all jobs. Ants start here and move throughout arcs considering direction constraints. If an ant comes toward a node, any other nodes that represent this process on other machines are deleted. When there is not any unvisited node in the graph, that means ant is finished its journey and has a solution for the IPPS problem. By using an equation that includes the next nodes' processes' makespan values and pheromone levels on arcs, an ant decides the next node to visit when it moves around. Pheromone level on arcs is key of this algorithm. In the first iteration, every node has the same pheromone level. But after the first iteration, the ant that obtained the best result increases pheromone levels of the arc that it's visited. Therefore, ants in the next iteration can use winner ants' path with higher percent.

3.2 Genetic algorithm (GA)

The hAG approach uses GA algorithm design which proposed in [25]. There are *job number* $\times$ *process number* $\times$ *2* genes in this chromosome model. In the first *process* $\times$ *job number* of a gene, genes show to that process must run that machine. In the second *process* $\times$ *job number*, genes show which job should be scheduled first in that solution.

For instance, when there are 2 jobs, 2 process and 2 machines;

$$[1\quad 2\quad 2\quad 1][2\quad 1\quad 2\quad 1]$$

chromosome means first job's first process must run 1st machine, 2nd process must run 2nd machine. 1st process of 2nd job must run 2nd machine, and 2nd process must run 1st machine. And first, 1st process of 2nd job must be scheduled, then 1st process of 1st job, 2nd process of 2nd job and finally 2nd process of 1st job. GA runs on this chromosome model.

The presented hAG hybrid approach basically uses one of the algorithms for improving results obtained by another. First, both algorithms run simultaneously and at the end of the

Genetic Algorithm Optimization	Ant Colony Optimization
Population: 100	Ant number: 10
Generation: 500	Max Iteration: 500
Mutation probability: 0.1	Max repeat number: 40
Crossover probability: 0.9	Trail evaporation rate: 0.5
Maximum repeat number: 50	Pheromone rate: 2
Selection method: Roulette Wheel	Desirability rate: 2
Mutation type: Swap Mutation	Initial pheromone rate: 2
Crossover type: Single point crossover	

Table 1. Hyperparameters of Genetic Algorithm Optimization and Ant Colony Optimization.

time limit, or any other stop condition, the supervisor detects which one gave better solutions. After this, the algorithm which has better solution continues and another one is stopped. If GA runs first, after it stops, ACO gets GA's good solutions as pheromone update on its graph. On the other hand, if ACO runs first after it stops, GA gets ACO's good solutions as individual chromosomes. Both ACO and GA have an avoid of convergence approach. So, if throughout a constant number of iterations, algorithms cannot find a better solution, parameters that cause them to stuck local optimum areas are reset. Therefore, the second algorithm has more solution to improve instead of one. Pseudocode of the algorithm is as presented below.

3.3 Proposed hybrid approach (hAG)

This proposed hybrid approach hAG basically uses one of those two algorithms GA and ACO for improving results initially obtained by the other. In this approach, the first phase is to select one of the appropriate methods, GA or ACO. Selection of the starting algorithm and the following algorithm is an important issue. The Algorithms' variable level of success in different problem types drives us to select a starting & following algorithm dynamically during running time.

First, both algorithms run simultaneously and at the end of the time limit, or any other stop condition, the supervisor detects which one gave better solutions. After this, the algorithm which had a better solution continues where the other one is stopped. If GA runs first; after it stops, ACO takes GA's good solutions as pheromone update on its graph. On the other hand, if ACO runs first after it stops, GA takes ACO's good solutions as individual chromosomes. Both ACO and GA have an avoid of convergence approach. Therefore, if throughout a constant number of iteration, algorithms cannot find a better solution, parameters that cause them to get stuck local optimum areas are reset. Hence, the second algorithm has more solutions for improvement instead of one. The pseudo code of hAG is as presented below.

```
Start
    timeLimit = 1000 ms
    solutionNumber = 10
    bestGA = GA(timeLimit)
    bestACO = ACO(timeLimit)
    Xrb = [ ]
    bestSolution = NEGATIVE INFINITY
    If (bestGA < bestACO)
        For (i in solutionNumber)
            Xrb.add(GA)
          For(i = 0 to solutionNumber)
            addPheromone(Xrb[i])
                solution = ACO( )
          If (solution < bestSolution)
            bestSolution = solution
          Else
            For (i in solutionNumber)
                Xrb.add(KKO)
          For(i = 0 to solutionNumber)
            addChromosome(Xrb[i])
          solution = GA( )
          If (solution < bestSolution)
            bestSolution = solution
      print(bestSolution)
  End
```

The inner mechanism and the hAG algorithm of the introduced approach are written in the pseudo codes given above. As mentioned before, the adaptive system performs according to the structure of the test case. The procedural steps in front of the proposed approach are handled due to the determinative mechanism, which decides which of the optimization approach is executed for the next step.

3.4 Artificial immune systems (AIS)

AIS is a rule-based machine learning systems inspired by the structure of the immune systems of living creatures. It is typically modeled due to the immune system's characteristics. AIS algorithm has been used in scheduling problems for more than 20 years. The basic approach is to create random antibodies that represent solutions and then trying to improve them with using various mutations. Antibody design of the algorithm is basically the same as the Genetic Algorithm. AIS does not take part in proposed hybrid approach but used as a comparison algorithm in chapter 4.3. The AIS algorithm in this paper constructed by using Engin & Doyen and Nhu Binh Ho and others' papers [26,27].

4. Experimental results

In experimental results, algorithms are compared in three problems, one of them was small and second was a large-scale problem. In these problems, each algorithm is executed 5 times to obtain a clearer result. In the third problem, the proposed hAG algorithm is compared with ACO, GA, and AIS with different 5 problems which include 5 jobs 4 processes and 3 machines. A table and a graphical method used for comparing algorithms. Table result uses three parameters for comparison. First is Success Rate (SR) and it indicates the percentage of all trials where the algorithm found the best solution. Next one is Average Relative Percentage Deviation (ARPD) and it shows how much percentage, algorithm makespan of solutions are worse than the best solution for that trial. The last parameter is how much Central Processing Unit (CPU) time algorithm is needed for running. In addition, a graphical method is used to compare algorithms. This graphic shows all better results found by the algorithm in any iteration. Hence, algorithms' progress with respect to time can be monitored with using this tool.

All algorithms are written in JavaScript and executed in Google Chrome browser in a MacBook Pro with 3 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3 memory.

4.1 Problem with 4 jobs – 4 processes – 4 machines

In this section, algorithms are tested in a problem that has 4 jobs, 4 processes, and 4 machines. All algorithms have been run 5 times. The average results are shown in Table 2. This hAG model and GA found best results in 60% of trials, ACO found best results in just 20% of trials. In addition, the hAG model's ARPD was a bit higher than GA and used more CPU time than other algorithms. Also, Figure 3 shows when each algorithm finds which solution exactly in

Table 2. Algorithm performance (computational time in s) table on a 4j/4p/4m problem.

Algorithms	Success Rate	ARPD	CPU
ACO	20	3.38	1.06
GA	60	0.75	1.76
hAG	60	2.77	3.37

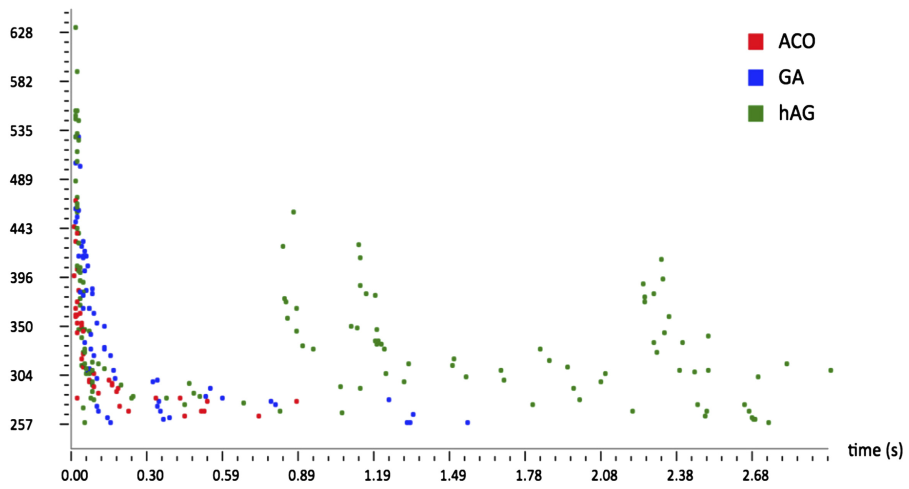


Figure 3.
Graphical results of a 4j/4p/4m problem.

all trials of this problem. ARPD and CPU are the Average Relative Percentage Deviation and Central Processing Unit respectively.

4.2 Problem with 6 jobs – 6 processes – 6 machines

In this section, all the presented algorithms are tested in a problem with 6 jobs, every job has 6 processes and processes have to be assigned in 6 machines. All algorithms executed 5 times and average results are shown in Table 3. In all evaluations, the hAG model has found the best solution. ARPD of GA was about 40% and ARPD of ACO was 14%. The hAG model significantly used more time than both two algorithms. In addition, in Figure 4, each algorithm's solution-finding times can be shown for all trials combined.

4.3 Shuffled 5 problems with ACO-GA-AIS and hAG

In this section, algorithms are compared with the hAG. These algorithms are tested with 5 different problems with 5 jobs, 4 operations, and 3 machines. Operations' running times of machines are shuffled among problems. Results can be seen in Table 4. The hAG model has found the best solutions in all tests but used more time than any other algorithms.

5. Conclusion

In this paper, a hybrid Ant Colony Optimization (ACO) - Genetic Algorithm (GA) approach is presented. This introduced hAG model has better a performance rate than other algorithms in

Algorithms	Success Rate	ARPD	CPU
ACO	0	13.38	7.38
GA	0	39.82	6.44
hAG	100	0	17.7

Table 3.
Algorithm success table (computational time in s) on a 6j/6p/6m problem.

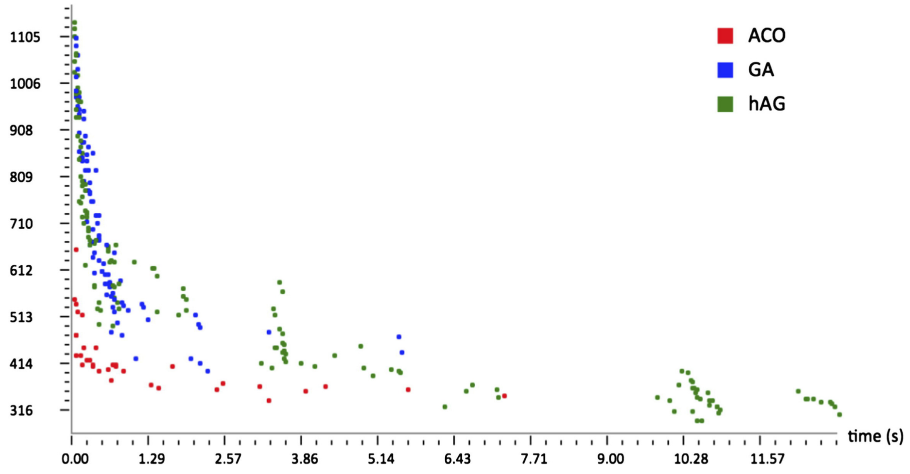


Figure 4.
Graphical results of a 6j/6p/6m problem.

Table 4.
Algorithm success table on (computational time in s) 5 different problems.

Algorithms	Success Rate	ARPD	CPU
ACO	20	5.54	1.02
GA	40	2.74	0.49
AIS	0	29.78	0.21
HAG	100	0.00	2.71

large-scale test problems. Also in small-scale test cases, this introduced model has a similar success rate (SR) with the genetic algorithm but better than ant colony optimization. In both types of problems, the proposed hybrid approach needs more CPU time for execution. Additionally, this study has experimentally proven that in large-scale problems ant colony optimization is better than the genetic algorithm and in small-scale problems, the genetic algorithm is better than ant colony optimization.

6. Availability

The presented IPPS study can be publicly evaluated from the (http://mecan.in/ipps/) website. Additionally, the implemented JavaScript source codes of GA, ACO, AIS, and hAG can be publicly downloaded from the URL addresses (http://mecan.in/ipps/ga.js), (http://mecan.in/ipps/aco.js), (http://mecan.in/ipps/ais.js), (http://mecan.in/ipps/hag.js) respectively for examinations and further studies.

References

[1] A. Allahverdi, A survey of scheduling problems with no-wait in process, Eur. J. Oper. Res. 255 (3) (2016) 665–686.

[2] H. Wen, S. Hou, Z. Liu, Y. Liu, An optimization algorithm for integrated remanufacturing production planning and scheduling system, Chaos, Solitons Fractals 105 (2017) 69–76.

[3] O. Engin, A. Güçlü, A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems, Appl. Soft Comput. 72 (2018) 166–176.

[4] A. Allahverdi, T. Aldowaisan, “Better Heuristics for M-Machine No wait Flowshop with Total Completion Time Criterion”, in: The 6th Saudi Engineering Conference, 2002, 4, pp. 551–560.

[5] L. Tseng, Y. Lin, A hybrid genetic algorithm for no-wait flowshop scheduling problem, Int. J. Prod. Econ. 128 (2010) 144–152.

[6] J. Li, E.K. Burke, R. Qu, Integrating neural networks and logistic regression to underpin hyper-heuristic search, Knowl.-Based Syst. 24 (2011) 322–330.

[7] D.C. Araújoa, M.S. Nagano, A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem, Int. J. Ind. Eng. Comput. 2 (2011) 155–166.

[8] L. Bianco, P. Dell’Olmo, S. Giordani, Flow shop no-wait scheduling with sequence dependent setup times and release dates, INFOR 37 (1999) 3–19.

[9] S.I. Brown, R.G. McGarvey, J.A. Ventura, Total flowtime and makespan for a nowait m-machine flowshop with set-up times separated, J. Oper. Res. Soc. 55 (6) (2004) 614–621.

[10] I.A. Chaudhry, S. Mahmood, “No-wait Flowshop Scheduling Using Genetic Algorithm”, No-wait Flowshop Scheduling Using Genetic Algorithm, 3 (2012): 4–6.

[11] D. Prot, O. Bellenguez-Morineau, Tabu search and lower bound for an industrial complex shop scheduling problem, Comput. Ind. Eng. 62 (2012) 1109–1118.

[12] G. Kumar, S. Singhal, Genetic algorithm optimization of flowshop scheduling problem with sequence dependent setup time and lot splitting, Int. J. Eng. Bus. Enterprise Appl. (IJEBEA) (2013) 62–71.

[13] T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, A unified solution framework for multi-attribute vehicle routing problems, Eur. J. Oper. Res. 234 (2014) 658– 673.

[14] E. Pacini, C. Mateos, C.G. Garino, Distributed job scheduling based on Swarm Intelligence: a survey, Comput. Electr. Eng. 40 (2014) 252–269.

[15] R. Burdett, E. Kozan, Determining operations affected by delay in predictive train timetables, Comput. Oper. Res. 41 (2014) 150–166.

[16] R. Pugazhenthi, M.A. Xavior, Computation of makespan using genetic algorithm in a flowshop, Am. Eurasian J. Sci. Res. 9 (2014) 105–113.

[17] D. Laha, S.U. Sapkal, An improved heuristic to minimize total flow time for scheduling in the m-machine no-wait flow shop, Comput. Ind. Eng. 67 (2014) 36–43.

[18] S. Dey, I. Saha, S. Bhattacharyya, U. Maulik, Multi-level thresholding using quantum inspired meta-heuristics, Knowl.-Based Syst. 67 (2014) 373–400.

[19] K. Kianfar, S.M.T.F. Ghomi, A.O. Jadid, Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA, Eng. Appl. Artif. Intell. 25 (2012) 494–506.

[20] Xinyu Li, Liang Gao, Effective Methods for Integrated Process Planning and Scheduling ISBN-13: 978-3662553039, Springer Publishing, New York, 2018.

[21] Grace Yuh-Jiun Lin, James J. Solberg, Effectiveness of flexible routing control, Int. J. Flex. Manuf. Syst. 3 (3-4) (1991) 189–211.

[22] Y.K. Kim, K. Park, J. Ko, A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling, Comput. Oper. Res. 30 (8) (2003) 1151–1171.

[23] A.M. Aguirre, S. Liu, L.G. Papageorgiou, Optimisation approaches for supply chain planning and scheduling under demand uncertainty, Chem. Eng. Res. Des. 138 (2018) 341–357.

[24] C.W. Leung et al., Integrated process planning and scheduling by an agentbased ant colony optimization, Comput. Ind. Eng. 59 (1) (2010) 166–180.

[25] Pelin Alcan, M. Fatih Uslu, Hüseyin Basligil, A Meta-Heuristic Approach for IPPS Problem, in: Uncertainty Modelling in Knowledge Engineering and Decision Making: Proceedings of the 12th International FLINS Conference, 2016. pp. 778–784.

[26] O. Engin, A. Doyen, A new approach to solve hybrid flow shop scheduling problems by artificial immune system, Future Gener. Comput. Syst. 20 (6) (2004) 1083–1095.

[27] N.B. Ho, J.C. Tay, E.M.K. Lai, An effective architecture for learning and evolving flexible job-shop schedules, Eur. J. Oper. Res. 179 (2) (2007) 316–333.

Corresponding author
Mehmet Fatih Uslu can be contacted at: mfatih.uslu@rumeli.edu.tr