Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem

Moch Saiful Umam [a,*], Mustafid Mustafid [b], Suryono Suryono [c]

[a] Magister Program of Information System, School of Postgraduate Studies, Diponegoro University, Semarang 50241, Indonesia
[b] Department of Statistics, Faculty of Science and Mathematics, Diponegoro University, Semarang 50275, Indonesia
[c] Department of Physics, Science and Mathematics Faculty, Diponegoro University, Semarang 50275, Indonesia

ARTICLE INFO

Article history:
Received 31 March 2021
Revised 23 July 2021
Accepted 23 August 2021
Available online xxxx

Keywords:
Genetic algorithm
Tabu search
Makespan
Flow shop

ABSTRACT

This paper combines the tabu search process with a genetic algorithm by employing a new partial opposed-based as the population initialization technique to minimize makespan. Flow shop is a prominent variety in scheduling with various applications in numerous disciplines, including production. Its difficulty makes many researchers develop algorithms to solve it since it is categorized as an NP-hard problem. Some approaches have been developed to resolve production scheduling, especially evolutionary algorithms. The genetic algorithm becomes the most used algorithm in evolutionary to address production schedule because of its capability to produce good, fast, and efficient results to explore complex solution space (global search). However, it provides ineffective results when doing exploitation (local search) that is easily stuck in the optimum local area. Contrarily, tabu search has superiority in local search to help genetic algorithms not be trapped in a local optimum. By combining these two algorithms with the initialization scheme, a new algorithm that balances searches in resulting higher quality solutions is obtained. The proposed algorithm was tested using 120 problem instances to carry out the algorithm performance. The empirical results state that the developed algorithm can improve 115 solutions out of 120 instances than the six other hybrid algorithms.

1. Introduction

The coordination and control from all of the complex organization tasks, including resource allocation, become important for the management function (Pinedo, 2016) to deliver the products or services as fast as possible to consumers (Mustafid et al., 2018). Year by year, the organization tasks are changed and more complex. Therefore, the reorganization in the production process should be done in a shorter time (Li et al., 2021), especially in scheduling to adapt to the varying demand of customers and the market.

Flow shop scheduling is renowned and classified as an NP-hard optimization problem (Berlińska and Przybylski, 2021), and several algorithms were developed to overcome this problem (Lee and Loong, 2019). Most researchers in this field used evolutionary algorithm (EA), more generally, nature-inspired metaheuristics algorithm due to its superiority (Greiner et al., 2018). In metaheuristic algorithms, most of them are inspired mainly by the natural evolution process, swarm behavior, and physics law. This population-based procedure employs more than one candidate solution when performing the search process to keep the population diversity and prevent the solutions trapped in local optima (Katoch et al., 2021).

Based on kinds of literature, we can mention some famous population-based metaheuristic approaches to address flow shop

Abbreviations: FSSP, flow shop scheduling problem; TS, tabu search; AOA, arithmetic optimization algorithm; GA-TS, genetic algorithm tabu search; GA-ISCR, genetic algorithm with insertion search cut and repair; ACO, ant colony optimization; CQGA, co-evolutionary quantum genetic algorithm; HGSA, hybrid genetic simulated annealing; TMIG, tabu-mechanism improved iterated greedy; SA, simulated annealing; IG-JP, iterated greedy with jumping probability; DSOMA, discrete self-organizing migrating algorithm; IG, iterated greedy; GA, genetic algorithm; AE, average error; PI, percentage of increase; ARPD, average relative percentage deviation.

* Corresponding author.
E-mail addresses: itgov@yandex.com (M.S. Umam), mustafid55@gmail.com (M. Mustafid), suryonosur@gmail.com (S. Suryono).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2021.08.025
1319-1578/

are genetic algorithm (GA) (Yu et al., 2018), iterated greedy (IG) (Shao et al., 2017), the ant colony optimization algorithm (ACO) (Engin and Güçlü, 2018), tabu search by swapping the neighborhood sequence (Gao et al., 2013), simulated annealing (SA) developed by (Wei et al., 2018), and recently using search method inspired from math operator called the arithmetic optimization algorithm (AOA) that was developed by (Abualigah et al., 2021a). Among metaheuristic solution approaches, GA is the most popular these days (Bisht et al., 2021), which is inspired by the biological evolution process, very useful for solving search and optimization problems in large and continuous search spaces (Keskin and Engin, 2021). The GA algorithm excels in solving scheduling problems because of its capability to achieve good, fast, and efficient results for exploring complex solution spaces (Piroozfard et al., 2016). But have problem in local search ability -easy to get trapped when exploited local optimum area (early convergence) (Amirghasemi and Zamani, 2017). On the other hand, Tabu Search (TS) nominated as the best local search method designed to get out of the optimum local, increasing the global optimization capability (Grabowski and Wodecki, 2004). By hybridizing the advantage of some algorithms, we can improve the base algorithm's performance (Raidl et al., 2019).

GA performs three biological-inspired operators after initializing the population, such as selection, crossover, and mutation (Sivanandam and Deepa, 2008). The initialization generates individual or feasible solutions as input to the GA using the various method. It is a very crucial task since it can bring an impact to the convergence speed and affect the final solution quality significantly (Vlašić et al., 2019). Hence, choosing a proper technique for the initial population in GA is critical to achieving a near-optimal solution and reducing computation time, i.e., minimizing makespan (Kazimipour et al., 2013). The random method is most commonly used in initializing the population, but it improves the convergence time and may guide the results to trap in local optima (Pan et al., 2014). For this reason, many papers refine the initial solution using some techniques to improve the solution.

This work developed a hybridization method between GA and TS to minimize makespan by employing the new technique called partial opposed-based in initializing the population. The GA acts as the global search scheme, and the TS is used to search neighborhoods or act as a local search scheme. The primary contribution of this article can be pointed as follows:

- We propose a hybrid GA-TS algorithm by employing a new partial opposed-based population initialization technique to address the flow shop problem.
- We develop a new approach for the initial population in GA to produce a feasible solution called partial opposed-based.
- The proposed GA-TS is calibrated for Taillard by 120 benchmark problems.

The remainder of this writing structure follows this order: A brief overview of the research and used methodology is covered in Section 2, followed by Section 3, which represent the result of experiment. Then the discussion of the proposed incorporation of the GA and TS algorithms is available in Section 4, and in the end, there is Section 5 that show the conclusions also future work.

2. Material and methods

2.1. Related works

This section contains two aspects starting from the various techniques used for initializing population in GA, followed by some metaheuristic algorithms to solve flow shop. The problem with production scheduling is to arrange production tasks that are sent to the machine according to the sequence, how to arrange processing resources and the order of operations under certain criteria which aim to minimize makespan (Pang et al., 2020). Flow shop scheduling is a problem that attracts researchers to solve and one of the popular scheduling types regarding how to arrange the job sequence through the same machining process (Belabid et al., 2019), with some preconditions in flow shop are (Komaki et al., 2019):

1. There is no dispatching rule between operations of separated jobs.
2. Any operation of a job or task can only be done on a machine unit every time.
3. For all of the jobs, the operation can be started if it was completed at the previous stage.
4. The job processing time on a certain machine can be 0, and the process is continued to the next machine.
5. The machines sequence is the same for all jobs; the problem is finding the jobs sequence on the machine which minimizes makespan.
6. The travel time between successive machines is negligible.

Previous studies within flow shop scheduling problem scope left a lot of room for further research. The process in GA will be started with initializing the population. With the importance of this process, many techniques were used, such as quasi-oppositional (Rahnamayan et al., 2007), space transformation search (Wang et al., 2009), opposition-based (Li and Yin, 2013), and linear regression (Hassanat et al., 2018).

In addressing the flow shop, the heuristic method is not effective anymore because of more problem complexity, so it must use metaheuristics (Lee and Loong, 2019). Most of the literature focuses on a single algorithm rather than a memetic or hybrid scheme (Tosun et al., 2020). Simulated annealing was the first metaheuristic to solve flow shop (Osman and Potts, 1989). Followed by GA (Rahman et al., 2015), and TS which is popular as local search optimization (Ben Cheikh-Graiet et al., 2020). Since the hybridization concept offers a more robust method by combining two or more algorithms, combining the GA and TS is an alternative solution in addressing the flow shop.

From some algorithms that have been hybridized, especially metaheuristic algorithms to address optimization problems, there is GA that is already combined with TS using a scatter search mechanism (Glover et al., 1995). For makespan minimization, there are GA which is hybridized with insertion search cut and repair algorithm (GA-ISCR) (Tseng and Lin, 2010), GA with a quantum-inspired evolutionary algorithm called with co-evolutionary quantum genetic algorithm (CQGA) (Deng et al., 2015), and GA combined with simulated annealing (HGSA) (Wei et al., 2018). Besides GA, various types of metaheuristics were proposed to reduce makespan in flow shop such as (Ding et al., 2015) that develop local search approach using variable neighborhood search called tabu-mechanism with the improved iterated greedy algorithm (TMIIG), then iterated greedy with jumping probability (IG-JP) (Tasgetiren et al., 2017), and (Davendra and Bialic-Davendra, 2013) used an iteration called migration in their discrete version of the self-organizing migrating algorithm (DSOMA). Also, in recent years, the aquila optimizer (Abualigah et al., 2021b), which is inspired by aquila's behavior process when catching the prey used for a general optimization problem.

2.2. The proposed methods

This section described methodology including two ideas: first, the proposed hybrid GA-TS to address the flow shop problem,

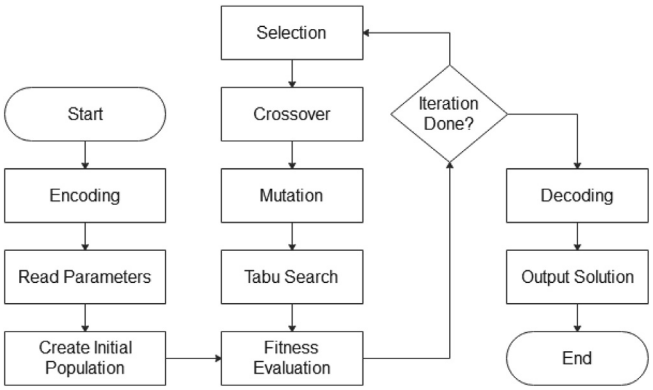

Fig. 1. Proposed GA-TS algorithm.

and second, the partial opposed-based technique to initialize the population. There are various ways to hybridize algorithms. In this paper, hybridization was carried out by inserting the TS algorithm into the GA procedure to explore a local search space. Fig. 1 below provides the workflow of the proposed GA-TS. In general, the developed algorithm can be written into phases, as shown below:Fig. 2.

Pseudo-code of proposed GA-TS
Phase 1: encoding the problem using job sequence Phase 2: parameter setting Phase 3: initialize the population, which can be done by partial opposed-based Phase 4: evaluate the fitness value Phase 5: is the maximal iteration reached? If reached, go to Phase 8 Else, go to Phase 6 Phase 6: reproduction of the new population Phase 6.1: reproduction using genetic operators (elitist and tournament selection, two-point crossover, swap mutation) Phase 6.2: run TS neighborhood (insert, swapping) Phase 7: go to Phase 4 Phase 8: chromosome decoding for the best solution

2.2.1. Parameter settings

A proper parameter tuning is needed to stabilize the proposed algorithm when exploring the solution and perform exploitation; Table 1 shows the adjusted parameters utilized for this study.

A hybrid GA-TS algorithm will stop the generation when the maximum iteration is reached. GA must stop iteration for itself also for TS, then decode the solution to show the optimal schedule. In order to get optimal results with acceptable computation time, a population size of 100 is used, and the maximum generation is 1000 (Yang et al., 2015). Once the population is generated, fitness value is evaluated, which is the objective function is to minimize makespan. Tournament size from (Kılıç and Yüzgeç, 2019) is increased from 2 to 5, that using a larger tournament size will produce a better solution. Crossover probability is 0,5 to ensure the "superior" parts of every algorithm are mixed toward a new solution, and mutation probability 0,1 is used to keep the population diversity (Yang et al., 2015).

Table 1
Parameter tuning.

No.	Parameter Name	Value
1	Population size	100
2	Maximum iteration	1000
3	Tournament size for selection	5
4	Probability of crossover	0.5
5	Probability of mutation	0.1
6	Length of tabu list	5

2.2.2. Initialize population

One of the good techniques for an initial population is opposed-based (Li and Yin, 2013), but when all solutions are positioned at the one side of search space, and the side is the property of local optima; then this state will convergence the algorithm too fast in the local optima. Hence, this can be tricked using our partial opposed-based by randomizing half solution and half another created by opposed-based to ensure all solutions do not fail in one side of search space, as we can see from the pseudo-code below:

Pseudo-code of partial opposed-based
1: Randomize population 2: Divide solution into two parts 3: Create half solution which is on the left part using opposition-based by comparing each solution S to S' and take the best fitness

2.2.3. Fitness evaluation

After creating the initial population, the fitness evaluation is performed by the fitness function or an objective function, which is in this paper is to make minimally possible the completion time of jobs (makespan, C_{max}), defined as:

$$f = minC_{max}$$

2.2.4. Choosing chromosome as parent

Selection is related to choosing the best chromosomes to serve as parents based on evaluating each chromosome's fitness value. In this study, two selection methods were used:

a. Elitist Selection

Every time you do a fitness evaluation, the chromosomes with the best fitness values are stored as an elitist solution so that in this study, the best fitness value, namely the smallest value, will be smaller or at least constant. This study adopts the elitist scheme from (Rani et al., 2019) with pseudo-code as follow.

Pseudo-code of elitist
1: P ← set of individuals within a population 2: OS ← offspring from reproduction 3: // combine individual in P and OS into elitist solution, TEMP 4: TEMP ← Merge (P, OS) 5: // copy best individual population size into P 6: P ← Copy Best (TEMP, P)

b. Tournament Selection

The tournament selection technique is considered more effective to minimize problems, so this study uses tournament size 5, considering that using a larger tournament size will produce a better solution (Lavinas et al., 2019).

Pseudo-code of tournament

```
P ← Population
i = individual
t ← size of the tournament, t ≥ 1
Best ← choose chromosome randomly from a population
For i to t do
Onward ← chromosome is chosen from a population
If Fitness (Onward) > Fitness (Best) then
Best ← Onward
Return Best
```

2.2.5. Reproduction process

The reproduction process is undertaken to obtain new chromosomes through crossover operator and mutation. This paper employed a two-point crossover, and the swap method is used in mutation.

a. Two-point crossover

Crossover is a combination of genes from selected solutions to create new offspring solutions. By exchanging part of a chromosome with another part of a chromosome, these tasks provide solution mixing and convergence in the search space. This paper using methods adopted (Vinoj and Jose, 2016) as follow.

Pseudo-code of two-point crossover

```
Choose two parent chromosomes randomly
Define two points as a boundary for changing chromosomes
Change the chromosomes that are between the two boundary points
```

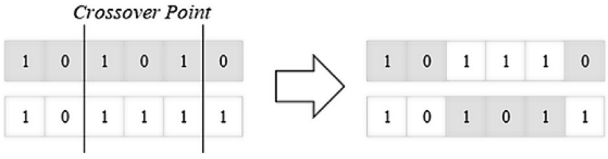

Fig. 2. Two points crossovers.

a. Swap mutation

This study uses swap mutation (Koohestani, 2020) by selecting two positions on chromosomes randomly and changing the values as shown in Fig. 3.

Pseudo-code of swap mutation

```
Choose two positions point in the population
Perform a swap to the element in the chosen position to produce offspring
```

2.2.6. Perform tabu search

TS has shown up to be the number one for local search strategies in scheduling problems which work based on the neighborhood search. The neighborhood framework from (Deroussi et al., 2012) is used in this study which consists of the move attribute, neighborhood structure, and tabu list. The searching steps started with the initial solution, followed by exploring the solution space; thus, TS moves from a solution to the next solution by selecting the best solution from the current neighborhood solution, which is not considered as a tabu solution or solution that has been explored and stored in a tabu list (Laguna, 2018).

Pseudo-code of tabu search process

```
Determine the initial solution and the length of tabu list
Determine the aspiration criteria, in this study, is make-span minimization
Determine the termination criteria; in this study, the iteration will stop if it has reached the maximum iteration of GA
Making a move, in this study using a neighborhood search with insertion and swapping methods
Evaluate the solution, update the solution with the best solution which is non-tabu of the search results
If the iterations have been maximal, then stop the search
```

2.2.7. Chromosomes decoding

The decoding method is used Job-Precedence (JP) carried by (Wang and Wang, 2019), which schedules the operation as a task sequence. It finishes the operation of job one on all machine, followed by the next job operation on all machines, and so forth. The pseudo-code used for decoding and displaying it in a Gantt chart:

Pseudo-code decoding algorithm

```
Initialize the Gantt chart construction
For every gene in chromosome do
Identify tasks of job J
Identify machine m to job J from the job starts with processing time
If J is the beginning job operation, Then
Set t0 = 0
Else
Set t0 to be time for J-1 job operation to finish
End If
End for
```

3. Results

This section brought the experimental computation to evaluate the proposed hybrid GA-TS. Using a 1.9 GHz CPU with 4 GB installed RAM and the dataset from Taillard, which contains 120 problems, performance investigations of the GA-TS have been carried out by testing ten times every Taillard problem instance separately fittest solution is recorded. Table 2 presents the summarized experiment results from the comparative analysis equipped with the average of error (AE), which is obtained using the formula:

$$AE = \frac{1}{K} x \sum_{i=1}^{k} \frac{GATS - Best}{Best} x100$$

K represents the number of instances, Best represents the fittest solution computed from the existing algorithm, and GA-TS represents the makespan or solution generated from the proposed algorithm. This experiment considers six hybrid algorithms: GA-ISCR, DSOMA, CQGA, TMIIG, IG-JP, and HGSA. The experiment results are:

Fig. 3. Swap mutation.

Table 2
Experiment results.

Instance	Size	Upper Bound	GA-ISCR	DSOMA	CQGA	TMIIG	IG-JP	HGSA	GA–TS	$\frac{GATS-Best}{Best} \times 100$
ta-001	20 × 5	1278	1449	1374	1486	1486	–	1324	1282	0,31
ta-002	20 × 5	1359	1460	1408	1528	1528	–	1442	1373	1,03
ta-003	20 × 5	1081	1386	1280	1460	1460	–	1098	1098	1,57
ta-004	20 × 5	1293	1521	1448	1588	1588	–	1469	1310	1,31
ta-005	20 × 5	1235	1403	1341	1449	1449	–	1291	1277	3,40
ta-006	20 × 5	1195	1430	1363	1481	1481	–	1391	1224	2,43
ta-007	20 × 5	1239	1461	1381	1483	1483	–	1299	1251	0,97
ta-008	20 × 5	1206	1433	1379	1482	1482	–	1292	1229	1,91
ta-009	20 × 5	1230	1398	1373	1469	1469	–	1306	1257	2,20
ta-010	20 × 5	1108	1324	1283	1377	1377	–	1233	1140	2,89
ta-011	20 × 10	1582	1955	1698	2044	2011	–	1713	1622	2,53
ta-012	20 × 10	1659	2123	1833	2166	2166	–	1718	1706	2,83
ta-013	20 × 10	1496	1912	1676	1940	1940	–	1555	1555	3,94
ta-014	20 × 10	1377	1782	1546	1811	1811	–	1516	1407	2,18
ta-015	20 × 10	1419	1933	1617	1933	1933	–	1573	1481	4,37
ta-016	20 × 10	1397	1827	1590	1892	1892	–	1457	1440	3,08
ta-017	20 × 10	1484	1944	1622	1963	1963	–	1622	1556	4,85
ta-018	20 × 10	1538	2006	1731	2057	2057	–	1749	1584	2,99
ta-019	20 × 10	1593	1908	1747	1973	1973	–	1624	1616	1,44
ta-020	20 × 10	1591	2001	1782	2051	2051	–	1722	1646	3,46
ta-021	20 × 20	2297	2912	2436	2973	2973	–	2331	2331	1,48
ta-022	20 × 20	2099	2780	2234	2852	2582	–	2280	2169	3,33
ta-023	20 × 20	2326	2922	2479	3013	3013	–	2480	2389	2,71
ta-024	20 × 20	2223	2967	2348	3001	3001	–	2362	2306	3,73
ta-025	20 × 20	2291	2953	2435	3003	3003	–	2507	2361	3,06
ta-026	20 × 20	2226	2908	2383	2998	2988	–	2375	2297	3,19
ta-027	20 × 20	2273	2970	2390	3052	3052	–	2341	2337	2,82
ta-028	20 × 20	2200	2763	2328	2839	2839	–	2279	2249	2,23
ta-029	20 × 20	2237	2972	2363	3009	3009	–	2410	2303	2,95
ta-030	20 × 20	2178	2919	2323	2979	2979	–	2401	2287	5,00
ta-031	50 × 5	2724	3127	3033	3161	3161	3002	2731	2730	0,22
ta-032	50 × 5	2834	3438	3045	3432	3432	3201	2934	2890	1,98
ta-033	50 × 5	2621	3182	3036	3211	3211	3011	2638	2622	0,04
ta-034	50 × 5	2751	3289	3011	3339	3339	3128	2785	2780	1,05
ta-035	50 × 5	2863	3315	3128	3356	3356	3166	2864	*2904*	1,43
ta-036	50 × 5	2829	3324	3166	3347	3347	3169	2907	2867	1,34
ta-037	50 × 5	2725	3183	3021	3231	3231	3013	2764	2755	1,10
ta-038	50 × 5	2683	3243	3063	3235	3235	3073	2706	2701	0,67
ta-039	50 × 5	2552	3059	2908	3072	3072	2908	2610	2601	1,92
ta-040	50 × 5	2782	3301	3120	3317	3317	3120	2784	2783	0,04
ta-041	50 × 10	2991	4251	3638	4274	4274	3638	3198	3100	3,64
ta-042	50 × 10	2867	4139	3511	4177	4177	3507	3020	3017	5,23
ta-043	50 × 10	2839	4083	3492	4099	4099	3488	3055	3015	6,20
ta-044	50 × 10	3063	4480	3672	4399	4399	3656	3124	3124	1,99
ta-045	50 × 10	2976	4316	3633	4322	4322	3629	3129	3123	4,94
ta-046	50 × 10	3006	4282	3621	4289	4289	3621	3293	3188	6,05
ta-047	50 × 10	3093	4376	3704	4420	4420	3696	3232	3226	4,30
ta-048	50 × 10	3037	4304	3572	4318	4318	3572	3390	3207	5,60
ta-049	50 × 10	2897	4162	3541	4155	4155	3532	3237	3045	5,11
ta-050	50 × 10	3065	4232	3624	4283	4283	3624	3251	3233	5,48
ta-051	50 × 20	3850	6138	4511	6129	6129	4500	4105	4064	5,56
ta-052	50 × 20	3704	5721	4288	5725	5725	4276	3992	3910	5,56
ta-053	50 × 20	3640	5847	4289	5862	5862	4289	3900	3875	6,46
ta-054	50 × 20	3720	5781	4378	5788	5788	4377	3921	3904	4,95
ta-055	50 × 20	3610	5891	4271	5886	5886	4268	4020	3929	8,84
ta-056	50 × 20	3681	5875	4202	5863	5863	4280	3971	3967	7,77
ta-057	50 × 20	3704	5937	4315	5962	5962	4308	4093	3968	7,13
ta-058	50 × 20	3691	5919	4326	5926	5926	4326	4090	3996	8,26
ta-059	50 × 20	3743	5839	4329	5876	5876	4316	4107	4064	8,58
ta-060	50 × 20	3756	5935	4422	5958	5958	4428	4113	3954	5,27
ta-061	100 × 5	5493	6492	6151	6397	6397	6151	5536	5502	0,16
ta-062	100 × 5	5268	6353	6064	6234	6234	6022	5302	5301	0,63
ta-063	100 × 5	5175	6148	6003	6121	6121	5927	5221	5213	0,73
ta-064	100 × 5	5014	6080	5786	6026	6026	5772	5044	5041	0,54
ta-065	100 × 5	5250	6254	6021	6200	6200	5960	5358	5323	1,39
ta-066	100 × 5	5135	6177	5869	6074	6074	5852	5197	5171	0,70
ta-067	100 × 5	5246	6257	6004	6247	6274	6004	5414	5320	1,41
ta-068	100 × 5	5094	6225	5924	6130	6130	5915	5130	5127	0,65
ta-069	100 × 5	5448	6443	6154	6370	6370	6123	5546	5506	1,06
ta-070	100 × 5	5322	6441	6186	6381	6381	6159	5480	5386	1,20
ta-071	100 × 10	5770	8115	7042	8077	8077	7042	5964	5962	3,33
ta-072	100 × 10	5349	7986	6813	7880	7880	6791	5596	5594	4,58
ta-073	100 × 10	5676	8057	6943	8028	8028	6936	5796	5790	2,01
ta-074	100 × 10	5781	8327	7198	8348	8348	7187	5928	*5939*	2,73

(*continued on next page*)

Table 2 (*continued*)

Instance	Size	Upper Bound	GA-ISCR	DSOMA	CQGA	TMIIG	IG-JP	HGSA	GA–TS	$\frac{GATS-Best}{Best} \times 100$
ta-075	100 × 10	5467	7991	6815	7958	7859	6810	5748	5637	3,11
ta-076	100 × 10	5303	7823	6685	7801	7801	6666	5446	5401	1,85
ta-077	100 × 10	5595	7915	6827	7866	7866	6801	5679	5667	1,29
ta-078	100 × 10	5617	7939	6874	7913	7913	6874	5723	5633	0,28
ta-079	100 × 10	5871	8226	6092	8161	8161	7055	5934	5926	0,94
ta-080	100 × 10	5845	8186	6990	8114	8114	6965	5998	5867	0,38
ta-081	100 × 20	6202	10,745	7854	10,700	10,700	7844	6395	6337	2,18
ta-082	100 × 20	6183	10,655	7910	10,594	10,594	7894	6433	6403	3,56
ta-083	100 × 20	6271	10,672	7825	10,611	10,611	7794	6689	6509	3,80
ta-084	100 × 20	6269	10,630	7902	10,607	10,607	7899	6419	6409	2,23
ta-085	100 × 20	6314	10,548	7901	10,539	10,539	7901	6536	6425	1,76
ta-086	100 × 20	6364	10,700	7921	10,690	10,690	7888	6527	6419	0,86
ta-087	100 × 20	6268	10,827	8051	10,825	10,825	7930	6542	6428	2,55
ta-088	100 × 20	6401	10,863	8025	10,839	10,839	8022	6712	6540	2,17
ta-089	100 × 20	6275	10,751	7969	10,723	10,723	7969	6760	6611	5,35
ta-090	100 × 20	6434	10,794	8036	10,798	10,798	7933	6621	6514	1,24
ta-091	200 × 10	10,862	15,739	13,507	–	15,319	13,406	11,120	11,103	2,22
ta-092	200 × 10	10,480	15,534	16,458	–	15,085	13,313	10,658	10,637	1,50
ta-093	200 × 10	10,922	15,755	13,521	–	15,376	13,416	11,224	11,220	2,73
ta-094	200 × 10	10,889	15,842	13,686	–	15,200	13,344	11,075	11,075	1,71
ta-095	200 × 10	10,524	15,692	13,547	–	15,209	13,360	10,793	10,756	2,20
ta-096	200 × 10	10,326	15,622	13,247	–	15,109	13,192	10,467	10,465	1,35
ta-097	200 × 10	10,854	15,877	13,910	–	15,395	13,598	11,394	11,174	2,95
ta-098	200 × 10	10,730	15,733	13,830	–	15,237	13,504	11,011	11,002	2,53
ta-099	200 × 10	10,438	15,573	13,410	–	15,100	13,310	10,725	10,721	2,71
ta-100	200 × 10	10,657	15,803	13,744	–	15,340	13,439	10,786	10,785	1,20
ta-101	200 × 20	11,195	20,148	15,027	–	19,681	14,912	11,642	11,528	2,97
ta-102	200 × 20	11,203	20,539	15,211	–	20,096	15,002	11,683	11,650	3,99
ta-103	200 × 20	11,281	20,511	15,247	–	19,913	15,186	11,930	*12,163*	7,82
ta-104	200 × 20	11,275	20,461	15,174	–	19,928	15,082	11,791	*12,098*	7,30
ta-105	200 × 20	11,259	20,339	15,047	–	19,843	14,970	11,728	*11,979*	6,39
ta-106	200 × 20	11,176	20,501	15,212	–	19,942	15,101	11,690	11,651	4,25
ta-107	200 × 20	11,360	20,680	15,168	–	20,112	15,099	11,958	11,957	5,26
ta-108	200 × 20	11,334	20,614	15,247	–	20,056	15,141	11,730	11,716	3,37
ta-109	200 × 20	11,192	20,300	15,136	–	19,918	15,034	12,138	12,120	8,29
ta-110	200 × 20	11,288	20,437	15,243	–	19,935	15,122	12,084	12,004	6,34
ta-111	500 × 20	26,059	49,095	37,064	–	46,689	35,372	26,859	26,771	2,73
ta-112	500 × 20	26,520	49,461	37,419	–	47,275	35,743	27,220	27,014	1,86
ta-113	500 × 20	26,371	48,777	37,059	–	46,544	35,452	27,511	27,491	4,25
ta-114	500 × 20	26,456	49,283	37,014	–	46,899	35,687	26,912	26,902	1,69
ta-115	500 × 20	26,334	48,950	36,894	–	46,741	35,417	26,930	26,790	1,73
ta-116	500 × 20	26,477	49,533	37,372	–	46,941	35,747	27,354	27,297	3,10
ta-117	500 × 20	26,389	48,943	36,698	–	46,509	35,395	26,888	26,758	1,40
ta-118	500 × 20	26,560	49,277	36,944	–	46,873	35,568	27,229	27,134	2,16
ta-119	500 × 20	26,005	49,207	36,862	–	46,743	35,304	28,103	27,636	6,27
ta-120	500 × 20	26,457	49,092	37,098	–	46,847	35,643	27,290	27,049	2,24

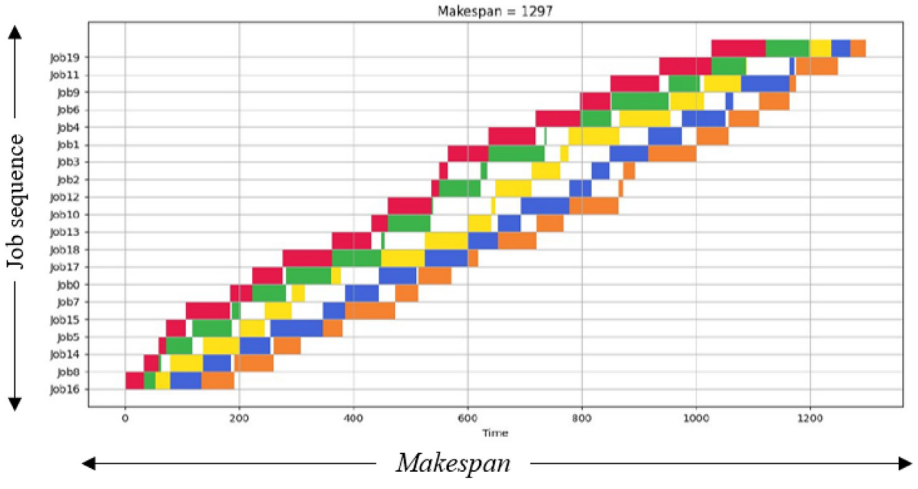

Fig. 4. Result of scheduling using GA-TS.

As an example, by entering the first problem Taillard 20 × 5, which means the problem has 20 jobs that are processed on 5 machines, the GA-TS algorithm has generated a schedule with job sequences [16, 8, 14, 5, 15, 7, 0, 17, 18, 13, 10, 12, 2, 3, 1, 4, 6, 9, 11, 19] and requires a completion time of 1,297 is given in Fig. 4.

As we can see in Table 2, the best makespan is recorded, and results from other algorithms are also presented there.

4. Discussion

We use three criteria to evaluate GA-TS makespan or solution quality: average error (AE), percentage of increase (PI) dan average relative percentage deviation (ARPD). Based on the computational result given in section 3, the rightmost column for the first problem, the GA-TS column, obtained *GATS* = 1282 and *Best* = 1278 so the column value is (*GATS* – *Best*) / *Best* × 100 = (1282 – 1278) / 1282 × 100 = 0.31 and the total is 366.09. From here, the AE value is obtained by:

$$AE = \frac{1}{120} x 366.09 = 3.05$$

The AE comparison from several other hybrid algorithms can be seen in Fig. 5, where the GA-TS has the lowest AE for the Taillard flow shop dataset.

The blue bars are algorithms other than GA-TS; the orange bars are GA-TS algorithms. The orange next to CQGA is the AE value of GA-TS for the first 90 examples of Taillard's problem, while the orange next to IG-JP is the AE GA-TS value for the last 90 examples of Taillard's problem.

The next analysis is the percentage of increase (PI), a relative change that shows the amount of increase in the final number to the initial number. In this case, PI represents the percentage increase in the GA-TS makespan compared to other algorithms. In the formula below C_{GATS} represents the minimum makespan produced using the GA-TS and C_x represents the minimum makespan generated from other algorithms being compared. As an example, C_{GATS} for problem size 20 × 5 is 12,441 and C_{GAISCR} for problem size 20 × 5 is 14265, then PI the GA-TS algorithm against GA-ISCR can be calculated with the following results:

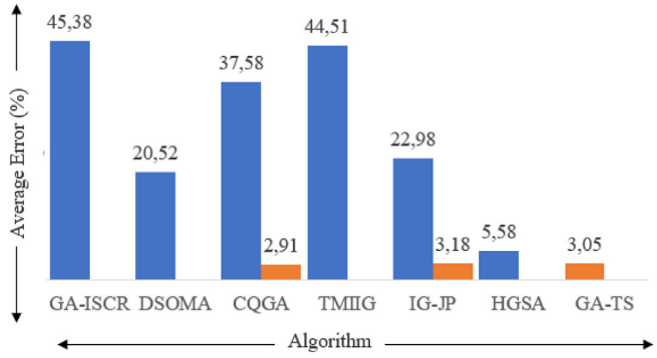

Fig. 5. AE comparison for some hybrid algorithms.

$$PI = \frac{C_{GAISCR} - C_{GATS}}{C_{GATS}} x 100$$

$$PI = \frac{14265 - 12441}{12441} x 100 = 14.66\%$$

The comparison of the percentage increase of the GA-TS algorithm with the other six algorithms can be summarized in Table 3. In the 200 × 20 problem, the PI value of the GA-TS algorithm recorded a negative value against the HGSA algorithm because, from the experimental results as shown in Table 2, this algorithm failed to minimize three problems: ta-103, ta-104, and ta-105, which resulted in the appearance of a minus sign. The best performance is obtained when solving a problem of size 100 × 20, and the GA-TS algorithm has improved the quality of the solution, which is visually presented in Fig. 6.

The last criterion is used to determine the average performance of GA-TS by employing average relative percentage deviation (ARPD), which can be described as the mean percentage value from the relative deviation among the results produced by the GA-TS with the optimal value. By adding up the difference in makespan for each algorithm with a known optimal value of makespan and then multiplying by 100 and dividing the result by 10, we get ARPD. Fig. 7 shows that the ARPD of the GA-TS algorithm has a value below 7% and is lower than the ARPD of other hybrid algorithms. The ARPD shows that the solution produced by the GA-TS can be close to the theoretical optimal value or called with an upper bound of the Taillard problem, lower is better.

Hybridization of GA and TS in every problem instance experiment is ranked as the best result for most flow shop problems. The first reason is that hybrid GA-TS is a combined algorithm between GA and TS. Both of them work together by balancing solution accuracy and computation speed to output a more optimal solution. The GA conducts exploration using its global search method; meanwhile, TS performs exploitation using its local search method. For the second reason, the main concern is to solve the flow shop effectively in terms of minimizing makespan located at the initial population technique and parameter settings used. Thus, GA-TS can find a solution by balancing the diversification (when performing global search) and intensification (when performing local search).

Table 3
GA-TS algorithm's percentage of increase.

Problem Size	Percentage of Increase GA-TS to					
	GA-ISCR	DSOMA	CQGA	TMIIG	IG-JP	HGSA
20 × 5	14,66	9,56	18,99	18,99	–	5,66
20 × 10	24,20	7,87	27,01	26,80	–	4,07
20 × 20	26,21	3,00	29,05	27,83	–	3,20
50 × 5	17,47	10,49	18,34	18,34	11,43	0,33
50 × 10	36,28	15,12	36,63	36,63	14,98	2,08
50 × 20	48,58	9,34	48,81	48,81	9,43	1,72
100 × 5	18,87	13,75	17,56	17,62	13,23	0,64
100 × 10	40,32	18,92	39,59	39,42	20,40	0,69
100 × 20	65,93	22,91	65,53	65,53	22,42	1,61
200 × 10	44,27	27,47	–	39,87	22,90	0,29
200 × 20	72,07	27,63	–	67,77	26,74	– **0,41**
500 × 20	81,51	36,77	–	72,82	31,19	0,54

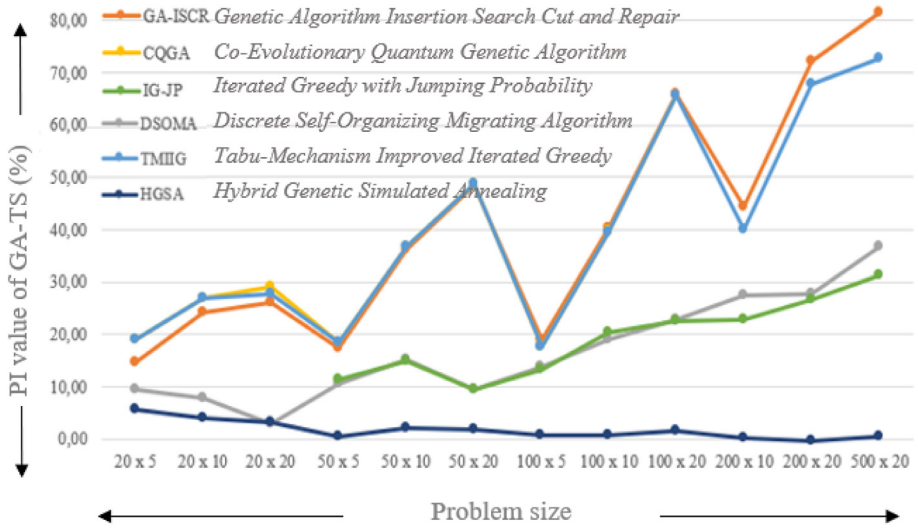

Fig. 6. PI value of GA-TS to another algorithm.

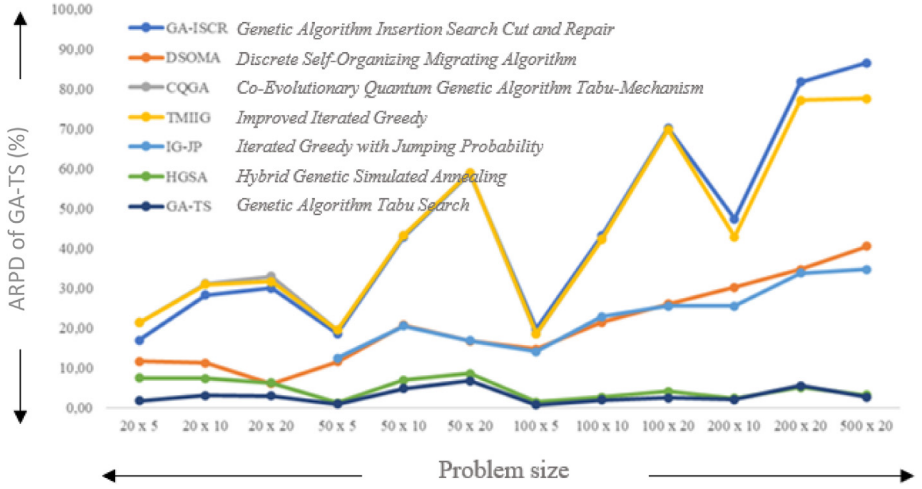

Fig. 7. ARPD value of GA-TS.

5. Conclusions

This research developed a hybrid GA-TS algorithm and successfully address makespan minimization in the flow shop. The three main genetic operators of the GA have been adapted, including the selection, crossover, and mutation, to solve flow shop effectively; the initial population technique becomes a success factor. Furthermore, the TS algorithm guides the local search for the GA algorithm to balance exploration and exploitation capabilities to find optimal solutions. 120 open problem instances from Taillard dataset, which is popular among researchers, have been utilized to evaluate the proposed hybrid GATS. We can state that the GA-TS has obtained a better increase in minimizing makespan in terms of solution accuracy from the conducted experiment compared to the other six hybrid algorithms. The GA-TS conducts AE value 3.05%; PI is varied and ARPD lower than 7%, also can improve 115 of 120 problem instances. This research can be expanded by combining the GA with another local search algorithm for future works under multi objectives, and multi constrain schema. Also, by determining dispatching rule can be implemented to achieve a more effective algorithm.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., Gandomi, A.H., 2021a. The arithmetic optimization algorithm. Comput. Methods Appl. Mech. Eng. 376, 113609. https://doi.org/10.1016/j.cma.2020.113609.

Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A.A., Al-qaness, M.A.A., Gandomi, A. H., 2021b. Aquila Optimizer: A novel meta-heuristic optimization algorithm. Comput. Ind. Eng. 157, 107250. https://doi.org/10.1016/j.cie.2021.107250.

Amirghasemi, M., Zamani, R., 2017. An effective evolutionary hybrid for solving the permutation flowshop scheduling problem. Evol. Comput. 25 (1), 87–111. https://doi.org/10.1162/EVCO_a_00162.

Belabid, J., Aqil, S., Allali, K., 2019. Solving flow shop problem with permutation and sequence independent setup time, in: 2019 International Conference on

Optimization and Applications, ICOA 2019. IEEE, pp. 1–5. https://doi.org/10.1109/ICOA.2019.8727667

Ben Cheikh-Graiet, S., Dotoli, M., Hammadi, S., 2020. A tabu search based metaheuristic for dynamic carpooling optimization. Comput. Ind. Eng. 140, 106217. https://doi.org/10.1016/j.cie.2019.106217.

Berlińska, J., Przybylski, B., 2021. Scheduling for gathering multitype data with local computations. Eur. J. Oper. Res. 294 (2), 453–459. https://doi.org/10.1016/j.ejor.2021.01.043.

Bisht, V.S., Joshi, N., Jethi, G.S., Bhakuni, A.S., 2021. A review on genetic algorithm and its application in power system engineering. Studies in Computational Intelligence., 107–130 https://doi.org/10.1007/978-981-15-7571-6_5.

Davendra, D., Bialic-Davendra, M., 2013. Scheduling flow shops with blocking using a discrete self-organising migrating algorithm. Int. J. Prod. Res. 51 (8), 2200–2218. https://doi.org/10.1080/00207543.2012.711968.

Deng, G., Wei, M., Su, Q., Zhao, M., 2015. An effective co-evolutionary quantum genetic algorithm for the no-wait flow shop scheduling problem. Adv. Mech. Eng. 7. https://doi.org/10.1177/1687814015622900

Deroussi, L., Gourgand, M., Norre, S., Deroussi, L., Gourgand, M., Norre, S., 2012. New effective neighborhoods for the permutation flow shop problem. Research Report. LIMOS/RR-06-09.

Ding, J.Y., Song, S., Gupta, J.N.D., Zhang, R., Chiong, R., Wu, C., 2015. An improved iterated greedy algorithm with a tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. Appl. Soft Comput. J. 30, 604–613. https://doi.org/10.1016/j.asoc.2015.02.006.

Engin, O., Güçlü, A., 2018. A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. Appl. Soft Comput. J. 72, 166–176. https://doi.org/10.1016/j.asoc.2018.08.002.

Gao, J., Chen, R., Deng, W.u., 2013. An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. Int. J. Prod. Res. 51 (3), 641–651. https://doi.org/10.1080/00207543.2011.644819.

Glover, F., Kelly, J.P., Laguna, M., 1995. Genetic algorithms and tabu search: Hybrids for optimization. Comput. Oper. Res. 22 (1), 111–134. https://doi.org/10.1016/0305-0548(93)E0023-M.

Grabowski, J., Wodecki, M., 2004. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Comput. Oper. Res. 31 (11), 1891–1909. https://doi.org/10.1016/S0305-0548(03)00145-X.

Greiner, D., Periaux, J., Quagliarella, D., Magalhaes-Mendes, J., Galván, B., 2018. Evolutionary algorithms and metaheuristics: applications in engineering design and optimization. Math. Probl. Eng. 2018, 1–4. https://doi.org/10.1155/2018/2793762.

Hassanat, A.B., Prasath, V.B.S., Abbadi, M.A., Abu-Qdari, S.A., Faris, H., 2018. An improved genetic algorithm with a new initialization mechanism based on Regression techniques. Inf. 9, 167. https://doi.org/10.3390/info9070167.

Katoch, S., Chauhan, S.S., Kumar, V., 2021. A review on genetic algorithm: past, present, and future. Multimed. Tools Appl. 80 (5), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6.

Kazimipour, B., Li, X., Qin, A.K., 2013. Initialization methods for large scale global optimization, in: 2013 IEEE Congress on Evolutionary Computation, CEC 2013. IEEE, pp. 2750–2757. https://doi.org/10.1109/CEC.2013.6557902

Keskin, K., Engin, O., 2021. A hybrid genetic local and global search algorithm for solving no-wait flow shop problem with bi criteria. SN Appl. Sci. 3, 628. https://doi.org/10.1007/s42452-021-04615-3.

Kılıç, H., Yüzgeç, U., 2019. Improved antlion optimization algorithm via tournament selection and its application to parallel machine scheduling. Comput. Ind. Eng. 132, 166–186. https://doi.org/10.1016/j.cie.2019.04.029.

Komaki, G.M., Sheikh, S., Malakooti, B., 2019. Flow shop scheduling problems with assembly operations: a review and new trends. Int. J. Prod. Res. 57 (10), 2926–2955. https://doi.org/10.1080/00207543.2018.1550269.

Koohestani, B., 2020. A crossover operator for improving the efficiency of permutation-based genetic algorithms. Expert Syst. Appl. 151, 113381. https://doi.org/10.1016/j.eswa.2020.113381.

Laguna, M., 2018. In: Handbook of Heuristics. Springer International Publishing, Cham, pp. 741–758. https://doi.org/10.1007/978-3-319-07124-4_24.

Lavinas, Y., Aranha, C., Sakurai, T., Ladeira, M., 2019. Experimental analysis of the tournament size on genetic algorithms, in: Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018. IEEE, pp. 3647–3653. https://doi.org/10.1109/SMC.2018.00617

Lee, T.S., Loong, Y.T., 2019. A review of scheduling problem and resolution methods in flexible flow shop. Int. J. Ind. Eng. Comput. 10, 67–88. https://doi.org/10.5267/j.ijiec.2018.4.001.

Li, X., Yin, M., 2013. An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Adv. Eng. Softw. 55, 10–31. https://doi.org/10.1016/j.advengsoft.2012.09.003.

Li, Y., Goga, K., Tadei, R., Terzo, O., 2021. Production scheduling in industry 4.0. Adv. Intell. Syst. Comput. 1194 AISC, 355–364. https://doi.org/10.1007/978-3-030-50454-0_34.

Mustafid, N.A., Karimariza, S.A., Jie, F., 2018. Supply chain agility information systems with key factors for fashion industry competitiveness. Int. J. Agil. Syst. Manag. 11 (1), 1. https://doi.org/10.1504/IJASM.2018.091352.

Osman, I.H., Potts, C.N., 1989. Simulated annealing for permutation flow-shop scheduling. Omega 17 (6), 551–557. https://doi.org/10.1016/0305-0483(89)90059-5.

Pan, W., Li, K., Wang, M., Wang, J., Jiang, B., 2014. Adaptive randomness: A new population initialization method. Math. Probl. Eng. 2014, 1–14. https://doi.org/10.1155/2014/975916.

Pang, X., Xue, H., Tseng, M.-L., Lim, M.K., Liu, K., 2020. Hybrid flow shop scheduling problems using improved fireworks algorithm for permutation. Appl. Sci. 10 (3), 1174. https://doi.org/10.3390/app10031174.

Pinedo, M.L., 2016. Scheduling: Theory, algorithms, and systems, fifth edition, Scheduling: Theory, Algorithms, and Systems, Fifth Edition. Springer London. https://doi.org/10.1007/978-3-319-26580-3

Piroozfard, H., Wong, K.Y., Hassan, A., 2016. A hybrid genetic algorithm with a knowledge-based operator for solving the job shop scheduling problems. J. Optim. 2016, 1–13. https://doi.org/10.1155/2016/7319036.

Rahman, H.F., Sarker, R., Essam, D., 2015. A genetic algorithm for permutation flow shop scheduling under make to stock production system. Comput. Ind. Eng. 90, 12–24. https://doi.org/10.1016/j.cie.2015.08.006.

Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A., 2007. Quasi-oppositional differential evolution, in: 2007 IEEE Congress on Evolutionary Computation, CEC 2007. IEEE, pp. 2229–2236. https://doi.org/10.1109/CEC.2007.4424748

Raidl, G.R., Puchinger, J., Blum, C., 2019. Metaheuristic hybrids. Int. Ser. Oper. Res. Manag. Sci. 272, 385–417. https://doi.org/10.1007/978-3-319-91086-4_12.

Rani, S., Suri, B., Goyal, R., 2019. On the effectiveness of using elitist genetic algorithm in mutation testing. Symmetry (Basel). 11 (9), 1145. https://doi.org/10.3390/sym11091145.

Shao, W., Pi, D., Shao, Z., 2017. Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms. Knowledge-Based Syst. 137, 163–181. https://doi.org/10.1016/j.knosys.2017.09.026.

Sivanandam, S.N., Deepa, S.N., 2008. Introduction to genetic algorithms, Introduction to Genetic Algorithms. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-73190-0

Tasgetiren, M.F., Kizilay, D., Pan, Q.K., Suganthan, P.N., 2017. Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput. Oper. Res. 77, 111–126. https://doi.org/10.1016/j.cor.2016.07.002.

Tosun, Ö., Marichelvam, M.K., Tosun, N., 2020. A literature review on hybrid flow shop scheduling. Int. J. Adv. Oper. Manag. 12, 156–194. https://doi.org/10.1504/IJAOM.2020.108263.

Tseng, L.-Y., Lin, Y.-T., 2010. A hybrid genetic algorithm for no-wait flowshop scheduling problem. Int. J. Prod. Econ. 128 (1), 144–152. https://doi.org/10.1016/j.ijpe.2010.06.006.

Vinoj, K., Jose, T., 2016. Flow shop scheduling using genetic algorithm. Int. J. Latest Trends Eng. Technol. 7 https://doi.org/10.21172/1.71.033.

Vlašić, I., Đurasević, M., Jakobović, D., 2019. Improving genetic algorithm performance by population initialisation with dispatching rules. Comput. Ind. Eng. 137, 106030. https://doi.org/10.1016/j.cie.2019.106030.

Wang, H., Wu, Z., Wang, J., Dong, X., Yu, S., Chen, G., 2009. A new population initialization method based on space transformation search, in: 5th International Conference on Natural Computation, ICNC 2009. IEEE, pp. 332–336. https://doi.org/10.1109/ICNC.2009.371

Wang, J. jing, Wang, L., 2019. Decoding methods for the flow shop scheduling with peak power consumption constraints. Int. J. Prod. Res. 57, 3200–3218. https://doi.org/10.1080/00207543.2019.1571252

Wei, Hongjing, Li, Shaobo, Jiang, Houmin, Hu, Jie, Hu, Jianjun, 2018. Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion. Appl. Sci. 8 (12), 2621. https://doi.org/10.3390/app8122621.

Yang, Xin-She, Chien, Su Fong, Ting, Tiew On, 2015. In: Bio-Inspired Computation in Telecommunications. Elsevier, pp. 1–21. https://doi.org/10.1016/B978-0-12-801538-4.00001-X.

Yu, C., Semeraro, Q., Matta, A., 2018. A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. Comput. Oper. Res. 100, 211–229. https://doi.org/10.1016/j.cor.2018.07.025.