Available online at www.dergipark.org.tr/en

INTERNATIONAL ADVANCED RESEARCHES
and
ENGINEERING JOURNAL

Journal homepage: www.dergipark.org.tr/en/pub/iarej

e-ISSN: 2618-575X

International Open Access

Volume 06
Issue 01

April, 2022

Research Article

Job shop scheduling with genetic algorithm-based hyperheuristic approach

Canan Hazal Akarsu [a,*] ***and Tarık Küçükdeniz*** [b]

[a] *Department of Industrial Engineering, Istanbul Esenyurt University, Istanbul, 34510, Turkey.*
[b] *Department of Industrial Engineering, Istanbul University – Cerrahpasa, Istanbul, 34320, Turkey.*

ARTICLE INFO

Article history:
Received 22 November 2021
Accepted 21 March 2022
Published 15 April 2022

Keywords:
Dispatching rules
GA-based hyperheuristics
Genetic algorithms
Hyperheuristics,
Job shop scheduling

ABSTRACT

Job shop scheduling problems are NP-hard problems that have been studied extensively in the literature as well as in real-life. Many factories all over the world produce worth millions of dollars with job shop type production systems. It is crucial to use effective production scheduling methods to reduce costs and increase productivity. Hyperheuristics are fast-implementing, low-cost, and powerful enough to deal with different problems effectively since they need limited problem-specific information. In this paper, a genetic algorithm-based hyperheuristic (GAHH) approach is proposed for job shop scheduling problems. Twenty-six dispatching rules are used as low-level heuristics. We use a set of benchmark problems from OR-Library to test the proposed algorithm. The performance of the proposed approach is compared with genetic algorithm, simulating annealing, particle swarm optimization and some of dispatching rules. Computational experiments show that the proposed genetic algorithm-based hyperheuristic approach finds optimal results or produces better solutions than compared methods.

1. Introduction

Job shop scheduling problem has been widespread and tough problem in literature that has been raised in recent 60 years [1]. There are many factories that make production worth millions of dollars with job shop type production system all around the world. It is really important to employ effective production scheduling methods in order to decrease costs and increase labor productivity and profitability [2]. The aim of this paper is to propose an effective GA-based hyperheuristic approach for JSSPs.

It has been accepted that the first studies, on which job shop scheduling was built, started in 1950s [3]. Johnson [4]'s algorithm, which was improved for double-machine flow type scheduling problem, was adapted to job shop scheduling problem by Jackson [5]. During 1950s, polynomial time algorithms directed towards the solution of the problem of job shop scheduling continued to be improved. The studies conducted in 1960s focused on reaching optimal result by using enumerative algorithms. Disjunctive graphs illustration was firstly proposed by Roy and Sussmann [6] in 1964 and the first disjunctive graph-based enumerative approach was implemented by Balas [7] in 1969. As well as being the most common enumerative approach, branch and bound algorithm's area of usage has been limited due to the fact that optimal solution can't be found for many problems. The complexity of problem has been the focal point of the studies during 1970s and the middle of 1980s. It has been discussed that only small parts of job shop scheduling problems could be solved with polynomial time algorithms and the majority of them belongs to NP-hard class. This situation has caused approximate solution methods to gain importance. Dispatching rules are the first approximate solution methods that have been improved. Between 1988 and 1991, innovative approximate solution methods [8] such as bottleneck algorithm and in 1990s, hybrid methods were developed [3]. While heuristic methods aim to produce appropriate solutions within reasonable calculation time, they don't guarantee optimal solution [9]. The reasons why the popularity of heuristic methods has increased since 1991 are the fact that their calculation power is strong and they have conversion features [10]. Dispatching rules, which are also called as priority rules or scheduling rules in literature, are the rules that are used for the solutions of scheduling problems because of its ease of implementation, quick reaction to sudden changes and production of valid solutions [11]. In literature, selection

* Corresponding author. Tel.: +90 444 9 123/1325; Fax: +90 212 699 09 90.
E-mail addresses: hazalakarsu@esenyurt.edu.tr *(C.H. Akarsu)*, tarik.kucukdeniz@gmail.com *(T. Küçükdeniz)*
ORCID: *0000-0003-1692-7727 (C.H. Akarsu), 0000-0002-6670-1809 (T. Küçükdeniz)*
DOI: 10.35860/iarej.1018604

of dispatching rules to be used is carried out with two methods: steady state simulation and artificial intelligence methods [11]. The selection of dispatching rules to be used within this study was provided by GA-based hyperheuristic approach.

Hyperheuristic approaches are high-level heuristics that look for a good solution instead of looking for a direct solution, by managing a series of low-level heuristics and they need limited problem-specific information [12]. The concept of hyperheuristic was first used to describe the protocol including the use of different artificial intelligence methods [13]. It was first described as "heuristics to choose heuristics" in accordance with combinatorial optimization as an independent concept in 2000 [14]. Burke et al. [13] divided hyperheuristics into two, according to the nature of heuristic search space. These are heuristic-selection that call for one of available low-level heuristics and heuristic-generation that form new heuristics by using available low-level heuristics [see 15–19]. Hyperheuristics can be classified depending on the type of low-level heuristics (LLHs) used as "construction" or "perturbation". Hyperheuristics that use dispatching rules as LLHs are included in construction class according to the structure of low-level heuristics. In this paper, GA-based hyperheuristic approach that is included in heuristic selection-construction class was proposed for job shop scheduling problems.

The aim of this paper is to propose an effective GA-based hyperheuristic approach for job shop scheduling problems. To the best of our knowledge, there are only few studies using GA-based hyperheuristic for the solution of the JSSP. This study is one of the few studies proposing a GA-based hyperheuristic for job shop scheduling problems. Other studies are given in Section 5. Unlike the few similar studies in the literature, 26 dispatching rules were used as low-level heuristics to provide an effective GA-based hyperheuristic approach for job shop scheduling problems.

2. Problem Statement

Job shop scheduling problem is described as qualifying starting time of each operation making objective function optimum by fulfilling precedence and capacity constraints specified in advance for performing n operations in m machines [20]. The number of possible schedules in JSSP is $(n!)^m$ for m machines and n operations [21]. In a study carried out by Lenstra and Rinnooy Kan [22], it was proved that JSSP was included in NP-hard class in case of availability of more than three machines. Relevant calculation time increases exponentially along with the increase in the size of the problem in order to obtain optimal solution for the problems that are included in NP-hard class. $J = \{J_1, J_2, \ldots, J_n\}$ is the set of jobs and $M = \{M_1, M_2, \ldots, M_m\}$ is the set of machines. It is accepted that each operation is performed only once in each machine. Processing of a job J_j in a machine M_i is called as an operation and the operation of a job is showed as O_{ij}. Each job follows special sequence of machines by technological constraints. Each job has its own sequence of machines and it is independent from the sequence of other jobs. Each O_{ij} operation has a special processing time and this time is shown as p_{ij}. All processing times are known and fixed. Setup times and transfer times are integrated into processing times.

In scheduling problems, a large number of complicated and sometimes conflicting objective functions can be built. Mellor [23] listed 27 scheduling problem objective functions. Minimization of maximum completion time is the first objective function used by the researchers in 1950s. Its academic and industrial use is common due to the fact that its mathematical formulation is easy [3]. Mathematical formulation of general job shop scheduling problem, which its objective function is the minimization of maximum completion time is as follows [21]:

C_{max}: maximum completion time (makespan)

y_{ij}: starting time of job j on machine i

p_{ij}: processing time of job j on machine i

N: set of all operations (i,j) that must be performed on n jobs.

A: set of all precedence constraints $(i,j) \rightarrow (k,j)$ that require job j to be processed on machine i before it is processed on machine k

min C_{max}

subject to

$$y_{kj} - y_{ij} \geq p_{ij} \; for \; all \; (i,j) \rightarrow (k,j) \in A \qquad (1)$$

$$C_{max} - y_{ij} \geq p_{ij} \; for \; all \; (i,j) \in N \qquad (2)$$

$$y_{ij} - y_{il} \geq p_{il} \; or \; y_{il} - y_{ij} \geq p_{ij} \; for \; all \; (i,l),(i,j) \in N \qquad (3)$$

$$y_{ij} \geq 0 \; for \; all \; (i,j) \in N \qquad (4)$$

Constraints (1) ensure all precedence constraints $(i,j) \rightarrow (k,j)$ that require job j to be processed on machine i before it is processed on machine k. Constraints (2) ensure that maximum completion time is not exceeded. Constraints (3) ensure that an operation cannot be interrupted, once it begins. Constraints (4) ensure nonnegativitiy of decision variables y_{ij}.

3. The GA Metaheuristic

The genetic algorithm (GA) developed by Holland [24] is a general search strategy and optimization method especially used in combinatorial optimization problems [25]. Genetic algorithms are used in many fields such as engineering, computer science, economy administration and supply chain management [26]. GA approach for the solution of scheduling problems was first developed by Davis [27] for job shop scheduling problem. GA has several advantageous features in comparison with other metaheuristics [26]:

- Flexibility of identification of constraints and quality measures,
- The ability to work with continuous and discrete variables,
- Ability of comprehensive search space,
- The ability to provide multiple optimal or good solutions,
- Use of parallel calculation techniques to reduce processing time.

Contrary to local search methods which handle one feasible solution such as simulated annealing and tabu search, GA uses population of solutions during search in order to prevent early convergence to local minimum [25]. Each individual in the population is called as a chromosome. Each chromosome consists of genes and represents a feasible solution. Chromosomes are evaluated with the calculation of fitness values. Crossover and mutation operators are used to produce new generations. Produced new generations are called as offspring. After many produced generations, the population converges with a solution that can be local or global optimum [28]. Since the time it was first developed, there have been significant improvements in chromosome representation of GA [29-31], crossover operator [32], mutation operator [32, 33], selection operator [32, 34] and generation of an initial population [35]. These developments have made traditional GA stronger [26]. Cheung and Zhou [36] proposed genetic algorithm and heuristic rule-based hybrid approach for JSSP. It was concluded that heuristic increased the performance of GA. Wang and Zheng [37] proposed integrated use of GA and SA for job shop scheduling problem. Zhou, Feng and Han [38] proposed integrated use of genetic algorithm, neighborhood search algorithm and dispatching rules (SPT, MWKR). It was stated that proposed algorithm proved good solutions in comparison with neighborhood search algorithm, simulated annealing and traditional GA. Gao, Sun and Gen [39] proposed integrated use of GA and variable neighborhood search methods for multi-objective flexible job shop scheduling problems. Variable neighborhood algorithm was used in order to increase convergence speed of GA. Two-vector chromosome representation that consists of a machine assignment vector and a job order vector was used in GA. It was shown that proposed method produced equal or better results in 157 out of 181 benchmark instances with available methods. Dao, Abhary and Marian [26] made a bibliometric analysis of published genetic algorithm studies between 1972 and 2014.

4. Dispatching Rules

Dispatching rules are the rules that are developed to make a quick selection among appropriate operations to be assigned in each time to form the schedule [18]. Dispatching rules are classified as static and dynamic based on the time variable [11]. While static dispatching rules such as earliest due date (EDD) and shortest processing time (SPT) can make evaluations before scheduling starts, dynamic rules such as most total work remaining (MTWR) and first in first out (FIFO) make changeable evaluations in any iteration while forming schedule [18]. According to their structural features, dispatching rules are classified as simple dispatching rules, combined dispatching rules, weighted dispatching rules and heuristic dispatching rules. Simple dispatching rules such as SPT, EDD have a single parameter objective function [11]. Simple dispatching rules can optimize limited scheduling objective functions [18]. Combined dispatching rules that are formed with the use of several simple dispatching rules are more successful for the solution of complicated problems. Weighted dispatching rules are linear combination of simple dispatching rules in which specified weights are assigned to operations according to their importance. Heuristic dispatching rules such as expert knowledge are generally used with simple, combined and weighted dispatching rules based on the structure of the system [11]. As new dispatching rules can be formed with the combination of one or more available dispatching rules, they can also be formed with one or more heuristic rules and dispatching rules [40].

Dispatching rules are heuristic rules that are often used for the solution of scheduling problems with the ability of producing valid solutions in limited time and quick response to sudden changes. When scheduling literature is considered, it is clear that sequencing rules, scheduling rules, priority rules and dispatching rules are used to describe the same technique [11]. Panwalkar and Iskander [41] listed and classified more than 100 dispatching rules. Dispatching rules have been used and continued to be developed for more than fifty years [41]. Even though there are many studies on dispatching rules, the majority of new studies are about combination and modification of old dispatching rules [11].

5. GA-based Hyperheuristic Algorithm

Generally, development of a heuristic, based on a relevant problem, is time consuming and it requires detailed information of the field. Though it gives successful results for the problems for which they are developed, they can’t be generalized to new or similar problems very often [42]. The motivation behind hyperheuristics is to expand application fields of developed methods. The studies carried out for this aim date back to the beginning of 1960s [13]. In early studies [see 43,44], when the concept of hyperheuristics wasn’t used yet, producing better results was aimed by using different dispatching rules in iterations according to a specific order.

Apart from simple low-level heuristics, hyperheuristics don't require any problem-specific information. Each LLH communicates with high-level heuristic with an interface structure independent from the problem. Hyperheuristic can call a LLH to see what happens when a LLH is used. Called LLH sends various parameters in which we can observe the performance of LLH such as objective function value and computational time [14].

Indirect chromosome representation is used in genetic algorithm-based hyperheuristics using genetic algorithm as high-level heuristic. In indirect chromosome representation, each chromosome represents the way of forming a solution instead of the solution itself [12]. In traditional GA, while chromosome directly encodes a solution via binary arrays or permutations, early studies were about indirect chromosome encoding as a result of difficulty of direct encoding due to the complicated nature of real-life problem [12]. Fang, Ross and Corne [45] is one of the first publications in which indirect chromosome representation is used. Eight dispatching rules were used as low-level heuristics for the solution of open shop scheduling problem. Each chromosome consists of a pair of gene sequences. In each pair, the first gene represents a dispatching rule and the second gene represents the job to be assigned with the implementation of this dispatching rule. It was seen that obtained results were very close to available solution in the literature; in fact, they were sometimes better.

To the best of our knowledge, there are only few studies using GA-based hyperheuristic for the solution of the JSSP. This study is one of the few studies proposing a GA-based hyperheuristic for job shop scheduling problems. Unlike the few similar studies in the literature, 26 dispatching rules were used as low-level heuristics in our study. Other studies are [43], [46] and [47]. Dorndorf and Pesch [43] used GA-based approach specifying sequence of LLHs to minimize makespan in JSSP. As low-level heuristics, twelve dispatching rules were used. One chromosome includes $n-1$ genes for n operations and each gene represents the heuristic to be used for scheduling an operation. Norenkov and Goodman [46] developed heuristic combination method for JSSPs. Scheduling problem was divided into two main problems as sequencing of jobs and their assignment to servers and heuristics were designated for each sub-problem. The aim of the method is to find optimal implementation order of heuristics. In its dimension, each chromosome is represented by a matrix of size $n \times q$ where n is the number of jobs and q is the number of consecutive service stages in which each job experiences during its processing. The schedule is formed for each service stage by adding a job in each stage. Vázquez-Rodríguez and Petrovic [47] developed hyperheuristic approach called as Dispatching Rule-Based GA for the solution of multiobjective JSSP. Genetic algorithm works in the space of dispatching rules and the chromosome representing the solution is expressed through (h, p) binary notation where h is the sequence of dispatching rules and p is the number of consecutive calls of each dispatching rule. The results of the approach were found to be superior when they were compared with GA using permutation chromosome representation and hyperheuristic using dispatching rule-based chromosome representation.

GA-based hyperheuristics are also used for other scheduling problems in [48-50]. Hart, Ross and Nelson [48] proposed GA-based approach for scheduling of chicken catching crew with the aim of sending chickens according to the orders received in chicken breeding facility, which is a real-life problem. A chromosome representation consisting of four parts was used. In the first part, there is a fixed problem specific criterion for narrowing the search space. In the second part, the information of the order sequence is included. In the third part, there is a heuristic dividing orders into duties and in the fourth part, there is a heuristic assigning the duties to the crew members. As a result of the study, it was observed that proposed method produced schedules within minutes competed with the schedules prepared by experts in days. For the trainer scheduling problem, Cowling, Kendall and Han [49] developed a GAHH approach called as hyper-GA. Working as a high-level heuristic, genetic algorithm specifies implementation sequence of LLHs into the problem. Twelve problem-specific heuristics were used as LLHs. In a chromosome, each gene was expressed through an integer value representing a LLH and the chromosome specifies implementation sequence of low-level heuristics. It was shown that proposed approach gave better results than the individual results of low-level heuristics and genetic algorithm. Bacha *et al.* [50] proposed a GA-based hyperheuristic approach for permutation flow shop problems. Authors used genetic algorithms as LLHs and also as a high-level heuristic. Each chromosome consisted of 10 genes and each gene includes a genetic operator or a parameter such as population size, number of generations, selection method, crossover method, crossover probability etc. The high-level heuristic was also a genetic algorithm which aim was to produce a tailored genetic algorithm to permutation flow shop scheduling problems. Since the effectiveness of GA depends on the selection of genetic operators and parameters, high-level GA calibrates low-level genetic algorithms. Authors indicated that there is definitely a correlation between benchmark instances and the appropriate GA parameters/operators that solve them efficiently. In the literature, GA-based hyperheuristic algorithms are also applied to different problems such as one-dimensional bin-packing problem [51], design of the packaging process of two-dimensional rectangular blocks

[52], multitask-oriented manufacturing service composition [53] and parameter tuning [54].

6. The Proposed GA-Based Hyperheuristic Algorithm

In this study, a genetic algorithm-based hyperheuristic approach is proposed for job shop scheduling problems. Proposed hyperheuristic approach is expressed through two modules: hyperheuristic module and low-level heuristics module. With a domain barrier, hyperheuristic module works independently from the problem. Hyperheuristics don't need to know problem-specific information or the names and methodologies of LLHs. LLHs are the heuristics directly working in the solution space of the problem. Therefore, LLHs module can be seen as a black box that hides problem details from the hyperheuristic and sends only objective function value of the current schedule to the hyperheuristic module.

6.1 Low-Level Heuristics Module

LLHs are the heuristics making a search in the solution space of the problem and requiring problem-specific information. In this paper, 26 dispatching rules shown in Table 1 were used as low-level heuristics. Dispatching rules were selected due to their success in scheduling problems, ease of implementation and flexibility [47]. The adaptation of developed approach to the changes in the problem can be easily provided by adding or removing new dispatching rules without changing the structure of the hyperheuristic.

The terminology below was used for the formulation of dispatching rules [55]:

C_{ij}	: completion time of operation j of job i
$N_{ij}(t)$	: number of jobs waiting in the queue containing operation j of job i at time t
p_{ij}	: processing time of operation j of job i
q	: index of unassigned operations ($q = j, \ldots, m_i$)
X_{ij}	: random number between 0 and 1 assigned to operation j of job i
$Y_{ij}(t)$	: total work of jobs waiting in the queue containing operation j of job i at time t
$Z_i(t)$	: priority value of job i at time t. Smallest values have greatest priority.

In traditional scheduling algorithm one dispatching rule is selected initially and assignments of all operations are performed according to this rule. In the proposed method, at each decision (assignment) point, the same or a different dispatching rule specified by the hyperheuristic is applied. Our developed scheduling algorithm is as follows:

1st step: Select the first unassigned operation according to job routing of each job and specify their machines. The precedence constraints of operations are satisfied in this step.

2nd step: Select the machine that becomes idle earliest among the machines.

3rd step: Assign the operation to selected machine according to the dispatching rule in k^{th} gene of the chromosome specified by the hyperheuristic ($k = 1$ in the first decision point).

4th step: Update schedule matrix. Update $k = k + 1$.

5th step: If all the operations are assigned, go to 6th step. Otherwise, go to 1st step.

6th step: Calculate objective function value of the schedule and send it to the hyperheuristic module.

6.2 Hyperheuristic Module

Hyperheuristic module is the part where the strategy of selecting the most appropriate heuristic to be called from LLH set is built in order to make the assignment at decision points of the problem. Steps of developed GA-based hyperheuristic module are as follows:

1st step: Create a population consisting of candidate solutions (each of them is called as a chromosome).

2nd step: Sent the population to LLH module.

3rd step: Select the chromosomes giving better results in accordance with objective function values returning from LLH module for the next population.

4th step: Create a new population from selected chromosomes with crossover and mutation operators.

5th step: Send the new population to LLH module again and take objective function values from LLH module.

6th step: If termination condition is not satisfied, go to the 3rd step. Otherwise, go to the 7th step.

7th step: Accept the chromosome giving the best objective function value so far as optimal result.

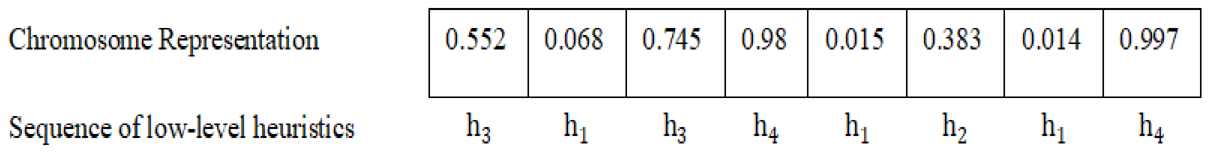
One of the factors specifying the quality of the solution is the method of encoding to chromosome. Cheng, Gen, & Tsujimura [56] stated that there were two main coding approaches of chromosome representation for job shop scheduling problems: direct encoding approach and indirect encoding approach. While in direct encoding approach, the schedule that is the direct solution of the problem is encoded onto the chromosome, in indirect encoding approach, for example in dispatching rule-based chromosome representation, implementation sequence of dispatching rules into the problem are encoded to the chromosome. In this paper, real valued dispatching rule-based chromosome representation is used. In dispatching rule-based representation, dispatching rules are matched with the numbers starting from 1 to the number of dispatching rules and candidate solution is created as a sequence of these numbers of size $m \times n$, where m is the number of machines and n is the number of jobs

Table 1. Low-level heuristics

DR	**Description** Selects the job which has…	$\boldsymbol{Z_i(t)}$
RANDOM	the smallest value of random priority	X_{ij}
FIFO	arrived at machine queue first	$C_{i,j-1}$
LIFO	arrived at machine queue last	$-C_{i,j-1}$
SPT	the shortest processing time (Also used as tie-breaking rule)	p_{ij}
LPT	the longest processing time	$-p_{ij}$
STPT	the shortest total processing time	$\sum_{j=1}^{m_i} p_{ij}$
LTPT	the longest total processing time	$-\sum_{j=1}^{m_i} p_{ij}$
LTWR	the least total work remaining	$\sum_{q=j}^{m_i} p_{iq}$
MTWR	the most total work remaining	$-\sum_{q=j}^{m_i} p_{iq}$
SDT	the smallest ratio of $(processing\ time \div total\ processing\ time)$	$p_{ij}/\sum_{j=1}^{m_i} p_{ij}$
LDT	the largest ratio of $(processing\ time \div total\ processing\ time)$	$-(p_{ij}/\sum_{j=1}^{m_i} p_{ij})$
SDR	the smallest ratio of $(processing\ time \div total\ remaining\ processing\ time)$	$p_{ij}/\sum_{q=j}^{m_i} p_{iq}$
LDR	the largest ratio of $(processing\ time \div total\ remaining\ processing\ time)$	$-(p_{ij}/\sum_{q=j}^{m_i} p_{iq})$
SMT	the smallest value of $(processing\ time \times total\ processing\ time)$	$p_{ij} \times \sum_{j=1}^{m_i} p_{ij}$
LMT	the largest value of $(processing\ time \times total\ processing\ time)$	$-(p_{ij} \times \sum_{j=1}^{m_i} p_{ij})$
SMR	the smallest value of $(processing\ time \times total\ remaining\ processing\ time)$	$p_{ij} \times \sum_{q=j}^{m_i} p_{iq}$
LMR	the largest value of $(processing\ time \times total\ remaining\ processing\ time)$	$-(p_{ij} \times \sum_{q=j}^{m_i} p_{iq})$
FRO	the fewest number of remaining operations	$m_i - j + 1$
LRO	the largest number of remaining operations	$-(m_i - j + 1)$
AVPRO_1	the shortest average processing time [59].	$\sum_{j=1}^{m_i} p_{ij}/m_i$
AVPRO_2	the longest average processing time [59]	$-(\sum_{j=1}^{m_i} \frac{p_{ij}}{m_i})$
SIO	shortest imminent operation [40].	$p_{i,j+1}$
WINQ	the least total work in the queue of its next operation	$Y_{i,j+1}(t)$
NINQ	the least number of jobs in the queue of its next operation	$N_{i,j+1}(t)$
PT+WINQ	the smallest value of $(processing\ time +$ total work in the queue of its next operation$)$[59].	$p_{ij} + Y_{i,j+1}(t)$
2PT+WINQ+NPT	the smallest value of $(2 \times processing\ time +$ total work in the queue of its next operation $+$ $processing\ time\ of\ next\ operation)$[60].	$2 \times p_{ij} + Y_{i,j+1}(t) + p_{i,j+1}$

In real valued representation, dispatching rules used as LLHs are represented by the number intervals between 0 and 1 instead of integers. A candidate solution consists of $m \times n$ unit of value, which is between 0 and 1. A low-level heuristic can be repeated more than once in a candidate solution.

An illustration of chromosome representation for a problem of size (2×4) is shown in Figure 1. Four low-level heuristics (h_1, h_2, h_3, h_4) were used and their intervals were defined by a minimum and a maximum value making selection probability of all low-level heuristics equal (25%). Each gene in the chromosome carries a random value. Each LLH falls into its specific discrete probability range and values in the chromosome are used to transform the chromosome representation to the LLH representation by using these discrete probability ranges. When the chromosome is decoded, the sequence of LLHs called by hyperheuristic respectively at each decision point is also given in Figure 1. It is seen that the solution obtained from hyperheuristic shows the way to be followed in order to reach optimal solution instead of direct solution of the scheduling problem.In this study, 26 low-level heuristics were used and their intervals were defined in order to make selection probability of all low-level heuristics equal. Until all operations were assigned, low-level heuristics encoded in the chromosome are called by hyperheuristic respectively at each decision point and an operation is assigned to the selected machine. Initial population is generated randomly. Uniform distribution is used as probability distribution function for creating the first population in real valued encoding.



Chromosome Representation	0.552	0.068	0.745	0.98	0.015	0.383	0.014	0.997
Sequence of low-level heuristics	h_3	h_1	h_3	h_4	h_1	h_2	h_1	h_4

Figure 1. Illustration of chromosome representation

Table 2. Average makespan of selected problems on benchmarked algorithms

Benchmark Instances	Optimum	Metaheuristics			Hyper-heuristic	Dispatching Rules				
		GA	SA	PSO	GAHH	SPT	MTWR	SIO	FIFO	PT+WINQ
FT06	55	56.96	56.20	57.96	55	84	58	69	74	74
LA01	666	672.43	666.76	677.56	666	817	828	867	912	912
LA06	926	926.13	926.00	926.00	926	1205	971	1199	1082	1082
LA11	1222	1226.26	1222	1225.76	1222	1537	1268	1453	1299	1299
LA19	842	939.63	899.06	942.50	876.93	1011	1000	1160	919	1028.7
ORB01	1059	1239.06	1173.26	1250.43	1085.46	1266	1374	1466	1327	1327
ABZ7	656	910.80	798.36	864.90	732.7	818	788	969	859	859
SWV11	2983	4920.83	4267.46	4745.23	3584.06	3712	4382	4232	4224	3923

In order to create a new population, some chromosomes coming from previous population should be selected to the new population as parent chromosomes. There are different methods in order to decide which chromosomes to choose. These methods are divided into two classes as fitness proportionate selection and ordinal selection [57]. In this study, roulette wheel selection, known as the most common proportional selection method, is used. In roulette wheel selection, roulette wheel slots are assigned to each individual proportional to its fitness value. Good solutions have more slots and therefore more probability of being selected. Roulette wheel selection steps are as follows [57]:

1st step: Calculate the fitness value (f_i) of each individual in the population.

2nd step: Calculate the probability of being selected (slot size) for each individual in the population: $p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$.

3rd step: Calculate the cumulative probability for each individual: $q_i = \sum_{j=1}^{n} p_j$.

4th step: Generate a uniform random number, $r \in (0,1]$.

5th step: If $r < q_1$ then select first chromosome (x_1), otherwise select chromosome x_i such that $q_{i-1} < r \leq q_i$.

6th step: Repeat 4th and 5th steps n times.

7. Computational Analysis

The performance of proposed GA-based hyperheuristic approach was evaluated by comparing with some metaheuristics and dispatching rules on a set of benchmark instances obtained from OR-Library [58]: FT06 (6×6), LA01 (10×5), LA06 (15×5), LA11 (20×5), LA19 (10×10), ORB1 (10×10), ABZ7 (20×15), SWV11 (50×10). The methods that were compared are GA, SA, PSO and SPT, MTWR, SIO, FIFO, PT + WINQ dispatching rules that were found to be successful in the literature for minimization of makespan in job shop problems.

In this paper, parallel GA and parallel GA-based hyperheuristic were used. Though genetic algorithms are more resistant against premature convergence to local optimum according to other local search methods, they aren't completely immune. One of proposed methods for reducing premature convergence of GA is parallelization of the GA into disjoint subpopulations. Parallel GA helps reducing computational time due to the use of parallel processors [25]. The code of proposed GAHH method was written in MATLAB. Population size, crossover rate and mutation rate parameter values for GA and GA-based hyperheuristic were used as 100, 0.7 and 0.1, respectively. The proposed GA-based hyperheuristic approach, GA, SA and PSO were run 30 times with 1000 iterations in each run. Average results are given in Table 2. The proposed GAHH approach gave optimal results for FT06, LA01, LA06 and LA11 instances and gave better results than the benchmark methods on the other instances. The best results for each run are given in Table 3. Also, boxplots of the 30 runs on each dataset can be seen in Figure 2. As an example, Gantt chart representing the optimal schedule to the FT06 instance is shown in Figure 3.

Table 3. Best values achieved among all the runs for each benchmarked algorithm

Benchmark Instances	Optimum	Metaheuristics			Hyper-heuristic	Dispatching Rules				
		GA	SA	PSO	GAHH	SPT	MTWR	SIO	FIFO	PT+WINQ
FT06	55	55	55	55	55	84	58	69	74	74
LA01	666	666	666	666	666	817	828	867	912	912
LA06	926	926	926	926	926	1205	971	1199	1082	1082
LA11	1222	1222	1222	1222	1222	1537	1268	1453	1299	1299
LA19	842	895	875	889	867	1011	1000	1160	919	1028.7
ORB01	1059	1171	1100	1183	1078	1266	1374	1466	1327	1327
ABZ7	656	857	767	820	723	818	788	969	859	859
SWV11	2983	4570	4128	4430	3513	3712	4382	4232	4224	3923

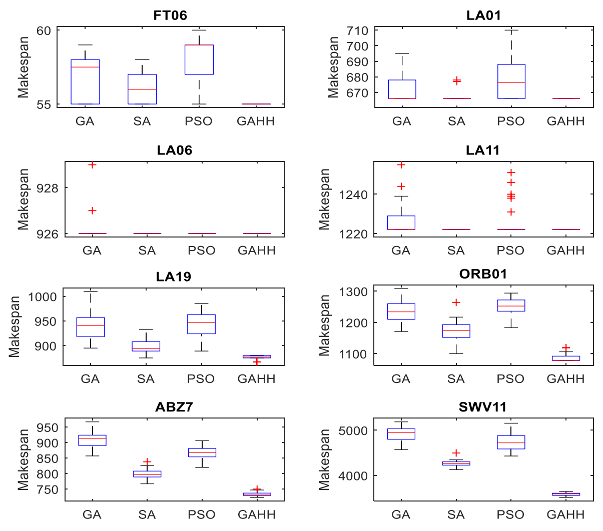


Figure 2. Boxplots of the 30 runs on each dataset

When average running results are considered, the proposed GA-based hyperheuristic approach gives the optimal results in 4 instances and produces better results than GA, SA, PSO, SPT, MTWR, SIO, FIFO, PT+WINQ in LA19 by 6.67%, 2.46%, 6.96%, 13.26%, 12.31%, 24.4, 4.58%, 14.75% respectively; in ORB01 by 12.4%, 7.48%, 13.19%, 14.26%, 21%, 25.96%, 18.2%, 18.2% respectively; in ABZ07 by 19.55%, 8.23%, 15.29%, 10.43%, 7.02%, 24.39%, 14.7%, 14.7% respectively; in SWV11 by 27.17%, 16.01%, 24.47%, 3.45%, 18.21%, 15.31%, 15.15%, 8.64% respectively.

When best running results are considered, the proposed GA-based hyperheuristic approach produces better results than GA, SA, PSO, SPT, MTWR, SIO, FIFO, PT+WINQ in LA19 by 3.13%, 0.91%, 2.48%, 14.24%, 13.3%, 25.26, 5.66%, 15.72% respectively; in ORB01 by 7.94%, 2%, 8.88%, 14.85%, 21.54%, 26.47%, 18.76%, 18.76% respectively; in ABZ07 by 15.64%, 5.74%, 11.83%, 11.61%, 8.25%, 25.39%, 15.83%, 15.83% respectively; in SWV11 by 23.13%, 14.9%, 20.7%, 5.36%, 19.83%, 16.99%, 16.83%, 10.45% respectively.

8. Conclusions

In today's highly competitive production environment, developing effective production scheduling methods has become a requirement for surviving in the market and maintaining competition. Increasing profitability is possible by increasing productivity in production and decreasing the costs by using effective scheduling methods. For this aim, a genetic algorithm-based hyperheuristic approach for job shop scheduling problem is proposed in this paper. The hyperheuristic approach is based on selection of the most appropriate heuristic from simple heuristics set which are called as low-level heuristics embedded under the hyperheuristic in order to make assignments at decision points of the problem. Since hyperheuristic approach doesn't need any problem-specific information except low-level heuristics.

In this paper, 26 dispatching rules were used as low-level heuristics. Dispatching rules were chosen due to their success, ease of implementation and flexibility in scheduling problems. The performance of proposed GA-based hyperheuristic approach was evaluated by comparing with GA, SA, PSO and SPT, MTWR, SIO, FIFO, PT + WINQ dispatching rules that were found to be successful in the literature for minimization of makespan in job shop scheduling problems. Eight benchmark instances in different sizes from different data sets were obtained from OR-Library. Computational experiments showed that the proposed GAHH approach has reached to the optimal solution in 4 out of 8 benchmark instances and was superior than the GA, SA, PSO and dispatching rules on the other instances. Computational results showed that proposed GAHH approach is an effective and promising method for JSSPs. We can further improve GAHH by combining a learning mechanism such as reinforcement learning. Reinforcement learning can avoid premature convergence and escape local optimum for 4 benchmark instances. Furthermore, we can test our proposed GAHH approach on different sets of benchmark instances and compare their results to the best methods in the state of art.

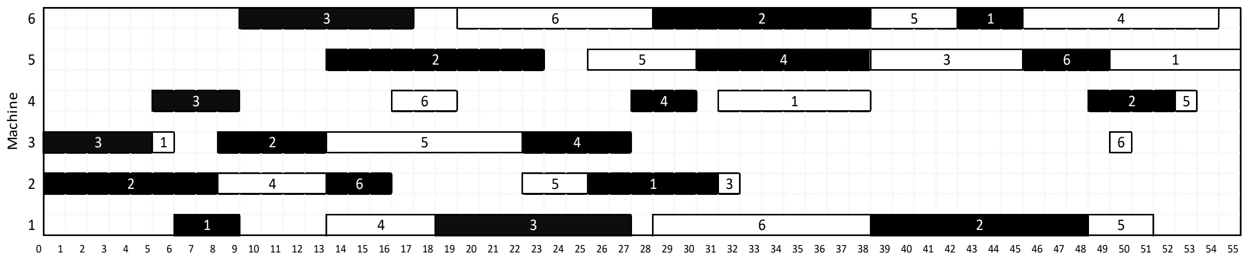


Figure 2. Gantt chart of optimal schedule for FT06

Declaration

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article. The authors also declared that this article is original, was prepared in accordance with international publication and research ethics, and ethical committee permission or any special permission is not required.

Author Contributions

C.H. Akarsu developed the methodology, performed the analysis and wrote the original draft. T. Küçükdeniz developed the methodology, supervised and improved the study.

Nomenclature

GAHH	: Genetic algorithm-based hyperheuristic
GA	: Genetic algorithm
SA	: Simulated annealing
PSO	: Particle swarm optimization
EDD	: Earliest due date
SPT	: Shortest processing time
MTWR	: Most total work remaining
FIFO	: First in first out
JSSP	: Job shop scheduling problem
LLH	: Low-level heuristic

References

1. Potts, C. N. and V. A. Strusevich, *Fifty years of scheduling : a survey of milestones*. J. Oper. Res. Soc., 2009. **60**(1): p. 41–68.
2. Jones, A., L. C. Rabelo, and A. T. Sharawi, *Survey of job shop scheduling techniques,* in *Wiley encyclopedia of electrical and electronics engineering,* 1999, Wiley Online Library.
3. Jain, A. S. and S. Meeran, *Deterministic job-shop scheduling: Past, present and future*. Eur. J. Oper. Res., 1999. **113**(2): p. 390–434.
4. Johnson, S., *Optimal two- and three-stage production schedules with setup times included*. Nav. Res. Logist. Q., 1954. **1**: p. 61–68.
5. Jackson, J., *An extension of Johnson's result on job-lot scheduling.* Nav. Res. Logist. Q., 1956. **3**(3): p. 201–204.
6. Roy, B. and B. Sussmann, *Les problemes d'ordonnancement avec contraintes disjonctives*. Note ds, 1964. **9**.
7. Balas, E., *Machine scheduling via disjunctive graphs: An implicit enumeration algorithm*. Oper. Res.,1969. **17**: p. 941–957.
8. Kovalev, M. Y., et al., *Approximation scheduling algorithms: A survey.* Optimization, 1989. **20**(6): p. 859–878.
9. Sharma, P. and A. Jain, *A review on job shop scheduling with setup times.* Proc. Inst. Mech. Eng. Part B J. Eng. Manuf., 2016. **230**(3): p. 517–533.
10. Jones, D. F., S. K. Mirrazavi, and M. Tamiz, *Multi-objective meta-heuristics: An overview of the current state-of-the-art.* Eur. J. Oper. Res., 2002. **137**(1): p. 1–9.
11. Fan, H. L., et al., *Survey of the selection and evaluation for dispatching rules in dynamic job shop scheduling problem,* in *2015 Chinese Automation Congress (CAC),* 2015. p. 1926–193.
12. Chakhlevitch, K. and P. *Cowling, Hyperheuristics: Recent developments* in *Adaptive and Multilevel Metaheuristics,* 2008, Springer. p. 3–29.
13. Burke, E. K., et al., *A classification of hyper-heuristic approaches,* in *Handbook of Metaheuristics,* 2010, Springer. p. 449–468.
14. Cowling, P., G. Kendall, and E. Soubeiga, *A hyper heuristic approach to scheduling a sales summit,* in *International conference on the practice and theory of automated timetabling*, 2020. p.176-190.
15. Hunt, R., M. Johnston, and M. Zhang, *Evolving machine-specific dispatching rules for a two-machine job shop using genetic programming,* in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014. p. 618–625.
16. Nguyen, S., et al., *Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming*. IEEE Trans. Evol. Comput., 2014. **18**(2): p. 193–208.
17. Sim, K. and E. Hart, *A novel heuristic generator for JSSP using a tree-based representation of dispatching rules,* in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 2015. p. 1485–1486.
18. Hart, E. and K. Sim, *A hyper-heuristic ensemble method for static job-shop scheduling.* Evol. Comput., 2016. **24**(4): p. 609–635.
19. Branke, J., et al., *Automated design of production scheduling heuristics: A review.* IEEE Trans. Evol. Comput., 2016. **20**(1): p. 110–124.
20. Yenisey, M. M. and M. Şevkli, *Atölye tipi çizelgeleme problemleri için parçacık sürü optimizasyonu yöntemi.* İTÜDERGİSİ/d, 2006. **5**(2): p. 58–68.
21. Pinedo, M. L., *Scheduling: Theory, algorithms and systems*. 2008, New Jersey: Prentice-Hall.
22. Lenstra, J. K. and A. R. Kan, *Computational complexity of discrete optimization problems.* Ann. Discret. Math., 1979. **4**: p. 121–140.
23. Mellor, P., *A Review of Job Shop Scheduling*. J. Oper. Res. Soc., 1966. **17**(2): p. 161–171.
24. Holland, J. H., *Adaptation in natural and artificial systems: an introductory analysis with application to biology, control, and artificial intelligence.* 1975, Ann Arbor (MI):

The University of Michigan Press.

25. Park, B. J., H. R. Choi, and H. S. Kim, *A hybrid genetic algorithm for the job shop scheduling problems*. Comput. Ind. Eng., 2003. **45**(4): p. 597–613.
26. Dao, S. D., K. Abhary, and R. Marian, *A bibliometric analysis of Genetic Algorithms throughout the history*. Comput. Ind. Eng., 2017. **110**: p. 395–403.
27. Davis, L., *Job shop scheduling with genetic algorithms,* in *Proceedings of an international conference on genetic algorithms and their applications*, 1985.
28. Gen, M. and R. Cheng, *Genetic algorithms and engineering optimization*. 2000, John Wiley & Sons.
29. Zhong, T. X. and J. C. Chen, *A hybrid-coded genetic algorithm based optimisation of non-productive paths in CNC machining*. Int. J. Adv. Manuf. Technol., 2002. **20**(3): p. 163–168.
30. Wang, Y. M., H. L. Yin, and J. Wang, *Genetic algorithm with new encoding scheme for job shop scheduling*. Int. J. Adv. Manuf. Technol., 2009. **44**(9–10): p. 977–984.
31. Dao, S. D., K. Abhary, and R. Marian, *Optimisation of partner selection and collaborative transportation scheduling in virtual enterprises using GA.* Expert Syst. Appl., 2014. **41**(15): p. 6701–6717.
32. Qing-dao-er-ji, R. and Y. Wang, *A new hybrid genetic algorithm for job shop scheduling problem.* Comput. Oper. Res., 2012. **39**(10): p. 2291–2299.
33. Tang, P.-H. and M.-H. Tseng, *Adaptive directed mutation for real-coded genetic algorithms.* Appl. Soft Comput., 2013. **13**(1): p. 600–614.
34. Wu, X., et al., *A genetic algorithm for cellular manufacturing design and layout.* Eur. J. Oper. Res., 2007. **181**(1): p. 156–167.
35. Šetinc, M., M. Gradišar, and L. Tomat, *Optimization of a highway project planning using a modified genetic algorithm.* Optimization, 2015. **64**(3): p. 687–707.
36. Cheung, W. and H. Zhou, *Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times.* Ann. Oper. Res., 2001. **107**(1–4): p. 65–81.
37. Wang, L. and D. Zheng, *An effective hybrid optimization strategy for job-shop scheduling problems.* Comput. Oper. Res., 2001. **28**(6): p. 585–596.
38. Zhou, H., Y. Feng, and L. Han, *The hybrid heuristic genetic algorithm for job shop scheduling.* Comput. Ind. Eng., 2001. **40**(3): p. 191–200.
39. Gao, J., L. Sun, and M. Gen, *A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems.* Comput. Oper. Res., 2008. **35**(9): p. 2892–2907.
40. Gere, W. S., *Heuristics in job shop scheduling*. Manage. Sci., 1966. **13**(3): p. 167–190.
41. Panwalkar, S. S. and W. Iskander, *A survey of scheduling rules.* Oper. Res., 1977. **25**(1): p. 45–61.
42. Burke, E. et al., *Hyper-heuristics: a survey of the state of the art.* J. Oper. Res. Soc., 2013. **64**(12): p. 1695–1724.
43. Dorndorf, U. and E. Pesch, *Evolution based learning in a job shop scheduling environment.* Comput. Oper. Res., 1995. **22**(1): p. 25–40.
44. Hart, E. and P. Ross, *A heuristic combination method for solving job-shop scheduling problems*, in *International Conference on Parallel Problem Solving from Nature,* 1998. p. 845–854.
45. Fang, H., P. Ross, and D. Corne, *A promising hybrid GA/heuristic approach for open-shop scheduling problems,* in *Proceedings of 11th European Conference on Artificial Intelligence*, 1994. p. 590–594.
46. Norenkov, I. P. and E. D. Goodman, *Solving scheduling problems via evolutionary methods for rule sequence optimization*, in *Soft computing in engineering design and manufacturing*, 1998, Springer. p. 350–355.
47. Vázquez-Rodríguez, J. A. and S. Petrovic, *A new dispatching rule based genetic algorithm for the multi-objective job shop problem.* J. Heuristics, 2010. **16**(6): p. 771–793.
48. Hart, E., P. Ross, and J. A. D. Nelson, *Scheduling chicken catching – An investigation into the success of a genetic algorithm on a real-world scheduling problem.* Ann. Oper. Res., 1999. **92**: p. 363–380.
49. Cowling, P., G. Kendall, and L. Han, *An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem*, in *the IEEE Congress on Evolutionary Computation (IEEE CEC02)*, 2002.
50. Ahmed Bacha, S. Z., et al., A *New Hyper-Heuristic to Generate Effective Instance GA for the Permutation Flow Shop Problem.* Procedia Comput. Sci., 2019. **159**: p. 1365–1374.
51. Ross, P., et al., *Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics,* in *Genetic and Evolutionary Computation Conference*, 2003. p. 1295–1306.
52. Thomas, J. and N. S. Chaudhari, *Design of efficient packing system using genetic algorithm based on hyper heuristic approach.* Adv. Eng. Softw., 2014. **73**: p. 45–52.
53. Zhang, S., Y. Xu, and W. Zhang, *Multitask-oriented manufacturing service composition in an uncertain environment using a hyper-heuristic algorithm.* J. Manuf. Syst., 2021. **60**: p. 138–151.
54. Pang, L. M., H. Ishibuchi, and K. Shang, *Using a Genetic Algorithm-based Hyper-heuristic to Tune MOEA/D for a Set of Various Test Problems,* in *IEEE Congress on Evolutionary Computation (CEC)*, 2021. p. 1486–1494.
55. Haupt, R., *A survey of priority rule-based scheduling.* OR Spectr., 1989. **11**(1): p. 3–16.
56. Cheng, R., M. Gen, and Y. Tsujimura, *A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation.* Comput. Ind. Eng., 1996. **30**(4): p. 983–997.
57. Sastry, K., D. E. Goldberg, and G. Kendall, *Genetic algorithms,* in *Search methodologies*, 2005, Springer. p. 97–125.
58. Beasley, J. E., *OR-Library*, 1990. [cited 2017 20 Nov]; Available from: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html.
59. Jayamohan, M. S. and C. Rajendran, *New dispatching rules for shop scheduling : A step forward*. Int. J. Prod. Res., 2000. **38**(3): p. 563–586.
60. Rajendran, C. and O. Holthaus, *Efficient jobshop dispatching rules: Further developments*. Prod. Plan. Control Manag. Oper., 2000. **11**(2): p. 171–178.