***EGE AKADEMİK BAKIŞ** / EGE ACADEMIC REVIEW*

Cilt 23 • Sayı 2 • Nisan 2023
SS. 237/246
Doi: 10.21121/eab.1246770
Başvuru Tarihi: 2.2.2023 • Kabul Tarihi: 20.2.2023

***Article Type:** Research Article*

Minimizing Makespan in a Permutation Flow Shop Environment: Comparison of Scatter Search, Genetic Algorithm and Greedy Randomized Adaptive Search Procedures

Ural Gökay ÇİÇEKLİ[1], Fatma DEMİRCAN KESKİN[2], Murat KOCAMAZ[3]

ABSTRACT

Solving scheduling problems enables more efficient use of production capacity. It involves defining the sequence of operations, determining the capacity of resources, and balancing workloads. Different methods, especially metaheuristics, have been used to solve these problems. This study presents the application of Scatter Search (SS), Genetic Algorithm (GA), and Greedy Randomized Adaptive Search Procedures (GRASP) for minimizing makespan in a permutation flow shop environment. In this study, the performances of these methods are compared through various test problems in the literature and a real-life problem of a company operating in the automotive sector. Study comprises 48 jobs that must be planned within a day for eight consecutive operations. In cellular manufacturing, the sequence in which each job is processed in eight operations is the same. In solving permutation flow shop scheduling problems (PFSP), one of the NP-hard problems, meta-heuristic methods are widely applied due to their successful results. From this point of view, SS, GA, and GRASP are employed in this study, and their performances are compared.

Keywords: Scheduling, Permutation Flow Shop Scheduling Problem, Makespan, Scatter Search, Genetic Algorithm, GRASP.

JEL Classification Codes: M100, M110, C630

Referencing Style: APA 7

INTRODUCTION

Scheduling is the process of planning and coordinating the production of goods or services to meet customer demand and achieve specific goals or objectives. It involves determining the sequence of operations, identifying and allocating resources (such as people, equipment, and materials), and establishing deadlines for completing each task. Production scheduling is important because it helps to coordinate the production process, allocate resources effectively, and meet customer demand in a timely and efficient manner. It also helps reduce waste, improve production efficiency, reduce inventory costs, and avoid production delays and bottlenecks. Additionally, it enables better communication and collaboration between different departments and ensures that the production process is aligned with the overall business strategy. Scheduling problems, which are combinatorial optimization problems, involve finding a schedule or assigning tasks to resources that optimize some objective function, subject to certain constraints. The higher the complexity, the harder it is to find the optimal solution to the problem (Widmer et al., 2008). Production scheduling problems have widespread coverage in the literature and have been addressed using different production systems, restrictions, source types, and objective functions. This research focuses on a particular type of scheduling problem, called the "PFSP", which is a subset of the more general class of scheduling problems known as "flow shop scheduling problems". The study applies this concept to a real-world scenario in the automotive industry, examining a specific company's operations.

PFSP contains a set of jobs that must be executed on a set of machines. Each job's execution should start from the first machine and finish on machine m following the same route of machines (Bautista et al., 2012). The processing time of all jobs on all machines is known in advance and can not be negative. All jobs are assumed to be ready before processing, and the processing of each job cannot be stopped after the processing has started. Only one job can be assigned to a machine at any given time. For example, to begin to process a job on

[1] Department of Business Administration, Ege University, İzmir, Turkey, gokay.cicekli@ege.edu.tr

[2] Department of Business Administration, Ege University, İzmir, Turkey, fatma.demircan.keskin@ege.edu.tr

[3] Department of Business Administration, Ege University, İzmir, Turkey, murat.kocamaz@ege.edu.tr

This paper is an extended version of the paper published in the Proceedings Book of 16th Production Research Symposium, on 12-14 October 2016.

a machine, that machine should be available, and the job processing on the previous machine from that machine should be completed. The aim is to sequence these n jobs to optimize the objective function (Zhang and Wu, 2014).

FSP and PFSP are scheduling problems that involve allocating resources to a set of tasks under certain constraints. The FSP involves scheduling a jobs on a set of machines such that each job must be executed on each machine in a specific order. Every job undergoes processing on each machine only once. (Ben-Daya and Al-Fawzan, 1998). The PFSP is a more restrictive version of the FSP. In the PFSP, the job order must be the same on all machines (Yenisey and Yagmahan, 2014). This means that the job schedule on the first machine determines its schedule on all other machines by aiming the makespan minimization. Therefore, the objective of the scheduling is also to minimize the makespan and total completion time. FSP and PFSP are combinatorial optimizations and NP-hard problems, which are computationally challenging. NP-hard problems are generally considered computationally difficult because most problems don’t have efficient algorithms for solving them. For this reason, many metaheuristic algorithms have been proposed to find approximate solutions to these problems.

Various studies in the literature address PFSP using exact and approximate algorithms to minimize the makespan of all jobs. As the dimension and complexity of the problems increase, it becomes impossible to solve them within a reasonable time using exact algorithms that guarantee the optimal solution. Today, most studies use heuristic and meta-heuristic methods, yielding good results within a reasonable time. In the literature, the study by Johnson (1954), which examined a PFSP with n jobs processed on two machines, was followed by other studies that suggest various heuristic methods for solving a PFSP with three or more machines. One of these heuristic methods is the CDS algorithm proposed by Campbell et al. (1970) by generalizing Johnson’s (1954) algorithm to flow shop problems with m machine. Then, Gupta (1972) proposed three heuristic algorithms that address PFSP through makespan minimization. The NEH algorithm is a well-known heuristic algorithm for solving the PFSP. It was first proposed by Nawaz et al. (1983). In comparing the performances of heuristics for solving PFSP using Taillard’s (1993) benchmarks, the NEH algorithm was found to have the most outstanding performance in makespan minimization (Ruiz and Maroto, 2005).

Various studies in the literature address FSP and PFSP using meta-heuristic methods. This study used three meta-heuristic methods, SS, GA, and GRASP, to solve PFSP. The introduction part of the study contains a literature review of production scheduling and PFSP. The second section of this study introduces the methods of SS, GA, and GRASP and the studies in the literature using these methods to solve such problems. The third section presents the problem analyzed in this study with its details. In the fourth section, the results obtained by the application of the addressed methods are given. This section compared the methods’ performances using the various benchmarks (rec31, rec33, rec35) in the literature and then a real problem. Finally, this study finalizes by presenting the conclusions regarding the results.

APPLIED META-HEURISTIC METHODS

In the literature, meta-heuristic methods have received considerable attention in solving FSP. Meta-heuristic methods are approximate optimization algorithms designed to find near-optimal solutions to difficult optimization problems. They do not guarantee that the solutions they find are globally optimal, but they are often able to find high-quality solutions. Meta-heuristic methods are particularly useful for solving problems that are computationally intractable, such as NP-hard problems, for which no exact algorithm can find the optimal solution in polynomial time. They are also useful for solving problems with many variables or constraints complexity, where traditional optimization methods may struggle to find reasonable solutions. Besides, they perform an efficient search and find a solution in a much shorter time than the traditional methods since they do not search the search space. Today’s intensely competitive business environment drives companies to find quick and reasonable solutions instead of optimum but slow solutions (Kocamaz and Çiçekli, 2010). Therefore, faster scheduling solutions can significantly impact the efficiency and effectiveness of operations, overall profitability, and competitiveness. This section presents the applied meta-heuristic methods in this study, SS, GA, and GRASP, and studies that applied these methods for PSFP.

Scatter Search

SS is a powerful and versatile population-based method metaheuristic optimization algorithm. A population-based approach commences with an initial set of solutions referred to as the reference set. Then, it gradually enhances them through a process of combination and modification in each iteration. SS is designed to handle complex, high-dimensional problems with multiple objectives and constraints. SS flexibility can handle various optimization problems, including mixed integer and nonlinear problems. It has been applied to many problems, such as scheduling, logistics, and resource allocation (Çiçekli and Bozkurt, 2015). SS is a robust algorithm that can handle a wide range of problem characteristics and quickly adapt to different problem domains. Compared to other optimization methods, scatter search has a relatively simple implementation process and does not require much fine-tuning of parameters.

Additionally, it can be integrated with other optimization algorithms and methods to improve performance and find more accurate solutions. It finds reasonable solutions to these problems in a relatively short computation time. Unlike other evolutionary algorithms, SS combines solutions through strategic designs to create a new one instead of relying on randomization (Haq et al., 2007).

SS aims to create new solutions by combining reference solutions. The logic behind combining solutions is to create a new solution using a linear (convex or concave) combination of at least two reference solutions (Laguna and Marti, 2003). The new reference set is developed until SS is finished by deleting the old solutions.

The fundamental steps of the SS algorithm include:

1. Initialization: Initial population of solutions is created.
2. Solution combination: New solutions are generated by combining existing solutions in the reference set.
3. Solution evaluation: The quality of the new solutions is evaluated using a suitable objective function.
4. Solution selection: The best solutions are selected to form the new reference set.
5. Iteration: The process is done again till satisfying a stopping criterion.

Various studies use this algorithm to solve FSP and PFSP in the literature. Nowicki and Smutnicki (2006) addressed PFSP using a new algorithm they proposed by combining some properties of neighborhoods, SS, and path relinking. Bozejko and Wodecki (2008) considered an FSP that aims to minimize the completion time using an SS-based parallel algorithm. Finally, Çiçekli and Bozkurt (2015) developed a model using SS to solve PFSP. To test their model, they used the PFSP called rec41, consisting of 75 jobs and 20 machines, developed by Reeves (1995).

Genetic Algorithm

As an optimization technique, GA reflects the principles of natural evolution. Michalewicz (1992) considers GAs as stochastic optimization techniques that mimic genetic inheritance and Darwin's principles of natural selection. GAs are population-based algorithms inspired by nature (Goldberg, 1989). They rely on the biological evolution models, which simulate survival of the fittest among individuals. This principle provides a mechanism to search for near-optimal solutions without trying all potential solutions. GAs do not guarantee the optimal solution, as it is often not possible to guarantee that the global optimum will be found (Çiçekli and Kaymaz, 2016). However, they can often find reasonable solutions close to optimal, especially when the search space is large and complex. The independent evaluation of candidate solutions allows for parallel processing, which can significantly speed up the optimization process.

The procedure for implementing a GA to solve a problem can be summarized as follows (Goldberg, 1989):

1. Encoding: The first step is encoding the problem as a set of binary strings or chromosomes representing the potential solutions. Each bit or gene in the chromosome represents a decision variable or feature of the problem.
2. Initialization: A population of chromosomes is randomly generated to represent the initial set of solutions.
3. Evaluation: The fitness value of each chromosome is evaluated with a suitable objective function. The objective function should be defined such that it assigns a higher fitness value to chromosomes that represent better solutions.
4. Selection: The selection method is usually grounded in the concept of natural selection, wherein chromosomes possessing higher fitness values have a greater probability of being selected.
5. Crossover: Chromosomes are combined to generate new offspring through a process called crossover. Crossover involves exchanging genetic information between chromosomes to create new genetic diversity.
6. Mutation: A small probability of mutation is applied to the offspring, introducing random changes in the genetic information. This helps to maintain genetic diversity and preserve getting stuck in local optima.

7. Replacement: The new offspring replace the previous population, and the process is done again till satisfying a stopping criterion.

8. Decoding: The final solution is obtained by decoding the best chromosome in the final population back into the original problem space.

The first step of GA involves finding the proper encoding methodology suitable for the problem. In each scheduling problem, the encoding of the chromosomes, mutation methods, and representation methodology is entirely different. Therefore, finding the proper encoding method before applying a GA to a problem is essential. Permutation coding is the most suitable approach for problems related to ordering (Borovska, 2006).

There are various important genetic operators in permutation coding, including crossover, selection, mutation, and elitism. These operations are utilized on the starting generation to uncover improved solutions in subsequent generations. The Genetic Algorithm begins by selecting parent individuals from the population (Kocamaz et al., 2009). Then, operators are used to select suitable parents. The selection of the most suitable parents and the next generation's production depends on the crossover operators. In most GA methods, crossover operators use two parents to create offspring. The new offspring always use the fittest alternative, thus ensuring the best ordering compared to the current one (Huang et al., 1997). The mutation is another genetic algorithm operator. Mutation relies on a random search for the optimal solution, while the initial generation does not impact the solution. In permutation coding, a mutation happens by swapping the place of the genes in the chromosomes. After that, GA repeats, generating new offspring until the expected number of iterations is achieved.

GA is an approach known to be effective in solving combinatorial optimization problems like scheduling problems. Various studies are using GA in solving PFSP. These studies investigated the problem of minimizing the total finishing time (Ruiz et al., 2005), total tardiness (Vallada and Ruiz, 2010) and total earliness and tardiness (Schaller and Valente, 2013). In the literature of GA-based solutions to the FSP, Babaei et al. (2012) dealt with the issue of lot sizing and scheduling in a flow shop setup with n-products and m-machines, considering factors such as sequence-dependent setup times, inventory costs, and the expenses associated with production delays and product waiting. Shahsavari Pour et al. (2013) also tackled the FSP using a GA-based strategy aimed at minimizing completion time, total waiting time, and overall tardiness.

Grasp

GRASP is a metaheuristic optimization technique combining greedy search and randomization elements to find near-optimal solutions. The basic idea behind GRASP is to iteratively construct solutions by selecting the best candidate among a set of randomly generated options while incorporating a local search component to improve the solution's value. GRASP can be easily adapted to different problem domains and can be combined with other heuristics to form hybrid algorithms. GRASP is a fast and effective algorithm that has been shown to produce good results on a wide range of problems. The GRASP iteration consists of two stages: the construction phase and the local search phase. The adaptive randomized greedy function is utilized during the construction phase to generate a feasible solution. Then, this solution is improved using local search algorithms in the following local search phase (Feo and Resende, 1995).

At each iteration, all candidate elements are evaluated based on a greedy evaluation function, and a new element selected from the ground set is included in the solution. This process ends when a feasible solution is found. The greedy evaluation function usually represents the change occurring in the objective function value when a new element is incorporated into the partial solution (Resende and Ribeiro, 2014).

A greedy parameter α [0,1] is used in a greedy evaluation function. GRASP is adaptive because the effects of each remaining element on the objective function are updated to reflect the effect brought on by choosing the last element at each iteration. GRASP is randomized because elements are selected randomly from a restricted candidate list, including the best candidates (Shahul Hamid Khan et al., 2007).

Locally optimality of the solutions produced during GRASP's construction phase is not exact. Therefore, GRASP's second phase, local search, is applied to improve these solutions (Festa and Resende, 2002). The studies applying GRASP indicate that different approaches are adopted in the construction and local search phases. In this study, we used the procedure proposed by Feo and Resende (1995) in the construction phase and the block insertion proposed by Allahverdi (2003) in the local search phase. The procedures are given step by step below:

GRASP- The Steps of the Construction Phase

Step1 - Compute each job's objective function value separately

Step2 - Estimate the objective function's minimum and maximum values

Step3 - Calculate Width=[α x (Max-Min)]

Step4 - Calculate the threshold values of the Restricted Candidate List (RCL)

RCL={ min, min + width}

Step5 - Randomly choose a candidate element from the RCL

Step6 - Repeat steps 1-5 till all jobs' assignments are completed

GRASP- The Steps of the Local Search Phase

Step 1: Take the job sequence (π) obtained in the constructing phase

Step 2: Assign as K=1. Generate alternative sequences of the first two jobs in π. Save the sequence, which has the best objective function value, as the best solution

Step 3: Assign as k=k+1. Take the following two jobs in π. Insert the kth block, which includes these two jobs, to all steps of the existing optimal solution both as it is and by changing the sequence of the jobs in the block. Out of all the candidate solutions, save the one with the best objective function values as the best solution.

Step 4: Repeat Step 3 till assignments of all jobs in π are completed.

GRASP is particularly useful for problems with many possible solutions and can be computationally expensive to solve. GRASP can be employed in a range of scheduling scenarios to optimize objectives such as makespan, total flow time, or total tardiness. There are studies in the literature that address FSP and PFSP using GRASP. For example, Considering FSP to minimize the makespan and maximum tardiness, Shahul Hamid Khan et al. (2007) used a semi-greedy heuristic in the construction phase and block insertion and random insertion perturbation in the local search phase. On the other hand, Arroyo and de Souza Pereira (2011) used a multi-objective GRASP-based heuristic to solve PFSP. They aimed to simultaneously minimize the completion time, maximum tardiness, and makespan and total flowtime. They used an NEH-based heuristic in the construction phase and insertion, general pairwise interchange, and two jobs insertion neighborhood during the local search phase. Finally, Molina-Sánchez and González-Neira (2016) used GRASP to solve PFSP to minimize the total weighted tardiness.

PROBLEM

This study analyzed the scheduling problem of a company that manufactures different-size pressure plates, discs, and release bearings for automobiles. The company has been operating in Izmir for nearly forty years. For the company using cellular manufacturing technologies during the manufacturing phase, a manufacturing cell consisting of eight consecutive machines which produce 215-430 mm rigid and pre-damper hub-type discs with organic bearings was analyzed. A total of 48 jobs in the manufacturing program of the manufacturing cell at a randomly selected shift were used in the study to create a manufacturing plan. All 48 jobs are processed on eight machines in the cell in the same order with different processing times. For this reason, it is seen that the company has a PFSP problem. Table 1 presents the codes of these jobs and the processing times of the machines.

There is no return and repetition between the operations. As a result, processing times are known for all jobs, and no setup is necessary during the transition from one product to another.

APPLICATION

In evaluating the performances of meta-heuristic methods, the extent to which the solutions created by these methods are close to the optimal solution is crucial. Unfortunately, the optimal solution to the real-life problem used in this study is unknown, and it takes too long to obtain it. Therefore, it may be challenging to evaluate the performances of meta-heuristic methods in solving large-scale scheduling problems whose optimal solution cannot be estimated. Therefore, to assess the performances of the meta-heuristic methods used in this study, we solved the test problems called rec31, rec33, and rec35 proposed by Reeves (1995) since these test problems consisting of 50 jobs and ten machines were closest to the real-life problem used in this study in terms of scale. The correlation between test data and real-world data is crucial in securing the results' validity and dependability. A higher degree of similarity between the two leads to more precise and representative results.

We used the Analytic Solver Platform developed by Frontline Systems, which can work with Microsoft Office Excel, to apply the SS and GA methods. The GRASP algorithm was coded using the VBA language. We used

Table 1: Jobs in the Shift Production Schedule and Their Processing Times

	Machines in the Manufacturing Cell							
Job	**RPr-O1**	**RPr-O2**	**RPr-O3**	**RPr-O4**	**RPr-O5**	**RPr-O6**	**RPr-O7**	**RPr-O8**
J01	1’10″	2’20″	2’00″	2’10″	2’25″	3’55″	2’30″	0’45″
J02	0’50″	2’20″	1’30″	2’10″	2’45″	2’50″	1’40″	0’25″
J03	2’00″	1’40″	3’40″	1’30″	2’25″	3’55″	2’30″	0’45″
J04	2’00″	1’40″	3’40″	1’30″	2’25″	3’55″	2’30″	0’45″
J05	2’00″	1’40″	3’40″	1’30″	2’25″	3’55″	2’30″	0’45″
J06	0’50″	1’40″	1’30″	1’30″	2’45″	3’00″	1’40″	0’45″
J07	1’10″	2’20″	2’00″	2’10″	3’45″	3’55″	2’30″	0’45″
J08	1’30″	2’20″	3’20″	2’10″	5’50″	2’20″	1’30″	0’25″
J09	1’30″	2’20″	3’20″	2’10″	5’50″	2’20″	1’30″	0’25″
J10	2’00″	1’40″	3’40″	1’30″	2’25″	2’50″	1’40″	0’25″
J11	1’10″	2’20″	2’00″	2’10″	2’45″	3’55″	2’30″	0’45″
J12	1’10″	2’20″	2’00″	2’10″	2’45″	3’55″	2’30″	0’45″
J13	1’10″	2’20″	2’00″	2’10″	2’45″	3’55″	2’30″	0’45″
J14	0’50″	2’20″	1’30″	2’10″	5’30″	3’55″	2’30″	0’45″
J15	2’00″	2’20″	3’40″	2’10″	3’50″	3’55″	2’30″	0’45″
J16	2’00″	2’20″	3’40″	2’10″	3’50″	2’50″	1’40″	0’25″
J17	1’30″	2’20″	3’20″	2’10″	5’50″	2’50″	1’40″	0’25″
J18	0’50″	2’20″	1’20″	2’10″	3’50″	3’55″	2’30″	0’45″
J19	1’30″	2’20″	3’20″	2’10″	5’50″	3’00″	1’40″	0’45″
J20	1’10″	1’40″	2’00″	1’30″	5’30″	2’50″	1’40″	0’25″
J21	1’30″	2’20″	3’20″	2’10″	3’50″	3’00″	1’40″	0’45″
J22	1’30″	2’20″	3’20″	2’10″	3’50″	2’50″	1’40″	0’25″
J23	2’00″	2’20″	3’40″	2’10″	2’45″	2’50″	1’40″	0’25″
J24	1’10″	1’40″	2’00″	1’30″	3’50″	2’50″	1’40″	0’25″
J25	0’50″	2’20″	1’20″	2’10″	3’50″	2’50″	1’40″	0’25″
J26	0’50″	2’20″	1’20″	2’10″	3’50″	2’50″	1’40″	0’25″
J27	0’50″	2’20″	1’20″	2’10″	3’50″	3’00″	1’40″	0’45″
J28	1’10″	1’40″	2’00″	1’30″	4’45″	3’55″	2’30″	0’45″
J29	1’10″	1’40″	2’00″	1’30″	5’50″	3’00″	1’40″	0’45″
J30	1’30″	2’20″	3’20″	2’10″	2’25″	3’00″	1’40″	0’45″
J31	2’00″	1’40″	3’40″	1’30″	3’50″	2’50″	1’40″	0’25″
J32	0’50″	1’40″	1’20″	1’30″	4’45″	3’00″	1’40″	0’45″
J33	0’50″	1’40″	1’20″	1’30″	4’45″	3’00″	1’40″	0’45″
J34	0’50″	1’40″	1’20″	1’30″	4’45″	3’00″	1’40″	0’45″
J35	0’50″	1’40″	1’20″	1’30″	4’45″	3’00″	1’40″	0’45″
J36	1’10″	2’20″	2’00″	2’10″	5’50″	2’50″	1’40″	0’25″
J37	1’10″	1’40″	2’00″	1’30″	5’30″	2’50″	1’40″	0’25″
J38	1’30″	2’20″	3’20″	2’10″	4’45″	3’00″	1’40″	0’45″
J39	1’10″	1’40″	2’00″	1’30″	3’45″	3’55″	2’30″	0’45″
J40	1’10″	1’40″	2’00″	1’30″	3’45″	3’55″	2’30″	0’45″
J41	1’10″	1’40″	2’00″	1’30″	3’45″	3’55″	2’30″	0’45″
J42	0’50″	2’20″	1’20″	2’10″	2’25″	3’00″	1’40″	0’45″
J43	1’10″	2’20″	2’00″	2’10″	3’45″	3’00″	1’40″	0’45″
J44	1’30″	1’40″	3’20″	1’30″	3’45″	3’00″	1’40″	0’45″
J45	1’30″	1’40″	3’20″	1’30″	3’45″	3’55″	2’30″	0’45″
J46	0’50″	1’40″	1’20″	1’30″	4’45″	2’50″	1’40″	0’25″
J47	1’10″	1’40″	2’00″	1’30″	2’25″	2’50″	1’40″	0’25″
J48	1’10″	1’40″	2’00″	1’30″	2’25″	2’20″	1’30″	0’25″

Table 2: Performance Summary of SS, GA, and GRASP for the Problems rec31-rec33-rec35

	Problem	Opt.	Min	Mean	Max	Dev (%)
SS	**rec31**	50’45”	54’11”	55’09”	62’35”	6,77%
	rec33	51’54”	52’20”	53’38”	55’02”	0,83%
	rec35	53’47”	55’42”	56’48”	60’35”	3,56%
GA	**rec31**	50’45”	51’21”	51’30”	53’00”	1,18%
	rec33	51’54”	52’20”	52’29”	53’40”	0,83%
	rec35	53’47”	54’37”	54’39”	54’53”	1,55%
GRASP	**rec31**	50’45”	53’00”	53’40”	54’06”	4,43%
	rec33	51’54”	53’45”	54’01”	54’10”	3,56%
	rec35	53’47”	55’26”	56’00”	56’21”	3,07%

the procedure prosed by Feo and Resende (1995) during the construction of the algorithm and the block insertion procedure proposed by Allahverdi (2003) during the local search phase. For all problems, 0, 0.2, 0.4, 0.6, 0.8, and 1 were tested as parameter α in the GRASP algorithm and the best solutions obtained after ten repetitions were presented.

During the application of GA and SS, constraint precision was taken as 0.00001, and the convergence value was taken as 0.0001. In the models, the population size was taken as 300, the mutation ratio was taken as 0.15, the random seed was taken as 25, and the tolerance limit was taken as 0.01. The models were finished when no progress occurred after 60 seconds. The model was run 50 times for both methods.

Table 2 gives the findings from solving the test problems using SS, GA, and GRASP.

The best sequences obtained for rec31, rec33, and rec35 using SS, GA, and GRASP are given in Table 3.

Although the three methods used differed in the results of the test problems, they reached the same results in real-life problem. When the real-life problem is solved using SS, GA, and GRASP, the makespan time of all jobs was found to be 11830 seconds by all three methods. Two alternative arrays give the best solution with all methods. These series are as follows:

Table 3: Best Job Sequences of SS, GA, and GRASP for the Problems rec31-rec33-rec35

rec31	**SS:**	J02-J48-J16-J40-J49-J10-J39-J24-J11-J38-J23-J37-J06-J29-J36-J17-J41-J19-J45-J50-J08-J44-J21-J18-J32-J30-J46-J07-J34-J22-J35-J25-J28-J12-J31-J05-J14-J42-J04-J33-J26-J01-J27-J13-J09-J03-J47-J15-J43-J20
	GA:	J18-J16-J34-J05-J23-J48-J06-J46-J17-J08-J36-J49-J40-J35-J37-J10-J11-J38-J26-J24-J42-J31-J03-J04-J39-J02-J44-J41-J28-J15-J14-J29-J32-J30-J21-J09-J22-J12-J33-J07-J50-J25-J19-J45-J13-J20-J47-J43-J27-J01
	GRASP:	J18-J16-J06-J40-J23-J48-J14-J50-J37-J11-J49-J04-J39-J38-J46-J25-J10-J34-J31-J42-J09-J03-J41-J19-J17-J29-J47-J05-J15-J02-J32-J08-J44-J24-J12-J36-J30-J21-J26-J45-J22-J01-J35-J07-J28-J27-J13-J33-J43-J20.
rec33	**SS:**	J31-J07-J44-0J3-J39-J14-J47-J36-J40-J22-J04-J08-J10-J37-J19-J02-J18-J21-J45-J42-J20-J15-J27-J30-J48-J11-J05-J25-J32-J38-J26-J29-J46-J06-J43-J49-J41-J13-J01-J28-J23-J09-J33-J12-J50-J16-J34-J35-J24-J17
	GA:	J31-J7-J45-J14-J47-J36-J27-J34-J39-J3-J42-J22-J43-J40-J04-J41-J15-J1-J48-J32-J30-J29-J2-J25-J50-J18-J8-J46-J38-J10-J44-J11-J37-J21-J26-J6-J19-J23-J5-J35-J9-J28-J12-J13-J33-J20-J49-J16-J24-J17
	GRASP:	J31-J03-J42-J07-J37-J27-J34-J39-J36-J18-J48-J25-J47-J22-J38-J15-J46-J10-J44-J29-J43-J26-J21-J13-J04-J20-J05-J06-J32-J11-J08-J30-J49-J40-J41-J01-J12-J02-J35-J28-J24-J23-J50-J09-J33-J14-J19-J16-J45-J17
rec35	**SS:**	J13-J50-J40-J10-J38-J37-J42-J17-J36-J26-J12-J19-J39-J5-J14-J11-J22-J3-J46-J48-J47-J45-J25-J4-J21-J44-J49-J6-J27-J43-J2-J8-J35-J7-J9-J1-J15-J18-J41-J34-J33-J28-J23-J20-J32-J30-J16-J29-J24-J31
	GA:	13-J14-J40-J2-J39-J50-J42-J25-J12-J10-J3-J36-J29-J21-J9-J17-J23-J45-J4-J44-J19-J41-J26-J27-J6-J46-J5-J33-J20-J35-J47-J15-J8-J37-J11-J38-J43-J22-J16-J18-J34-J1-J30-J7-J48-J49-J32-J28-J24-J31
	GRASP:	J13-J2-J14-J29-J5-J47-J36-J40-J37-J38-J50-J42-J4-J26-J9-J46-J30-J22-J43-J10-J8-J33-J48-J3-J34-J39-J6-J41-J19-J23-J35-J20-J18-J21-J15-J12-J11-J27-J16-J1-J25-J32-J24-J17-J44-J45-J7-J31-J49-J28.

Table 4: Best Alternative Job Sequences of SS, GA, and GRASP for the Real-Life Problem

Real-Life Problem	**Alternative Sequence 1:**	J32-J14-J37-J27-J43-J23-J40-J33-J29-J19-J41-J18-J38-J48-J02-J17-J46-J30-J34-J45-J39-J24-J25-J4-J35-J3-J28-J10-J13-J44-J20-J16-J8-J5-J6-J26-J42-J12-J15-J47-J11-J7-J21-J31-J36-J1-J22-J9
	Alternative Sequence 2:	J32-J2-J12-J9-J3-J13-J31-J14-J29-J41-J37-J46-J26-J45-J11-J34-J39-J23-J20-J1-J15-J27-J19-J6-J42-J48-J4-J38-J24-J28-J21-J25-J30-J18-J33-J07-J43-J35-J40-J44-J05-J16-J17-J10-J47-J36-J22-J8

CONCLUSION

The use of meta-heuristics in scheduling within the manufacturing sector in Turkey is likely to have gained significant attention in recent years. As companies strive to improve their production processes, they are looking for effective scheduling methods that can help optimize their operations. Effective scheduling leads to improved productivity, increased customer satisfaction, better resource management, and reduced costs, making it an essential aspect of operations management and business success. As a result, meta-heuristics, with their ability to provide flexible, efficient, and high-quality solutions to scheduling problems, are becoming increasingly popular among Turkish manufacturing companies. This study addressed PFSP with a real-life problem of a company that adopted cellular manufacturing technologies. The PFSP, a widely researched topic in the field for many years, is recognized as an NP-hard problem. As a result, meta-heuristic techniques are often utilized to address this challenging issue. The PFSP was tackled using SS, GA, and GRASP, and the efficiency of these meta-heuristic methods was then compared to one another. Furthermore, to evaluate the performances of these methods, the test problems called rec31, rec33, and rec35 proposed by Reeves (1995) were solved since these test problems were closest to the real-life problem used in this study in terms of scale.

SS uses a set of reference solutions to guide the search process, GA uses genetic operations to evolve a population of solutions, and GRASP uses a greedy strategy combined with randomization to generate solutions incrementally. GA was the most successful method in solving all test problems. GRASP was more successful for rec31 and rec35 than SS; however, SS obtained the same result as GA for rec33, yielding a solution with only a 0.83% deviation from the optimal solution. The analysis of the methods' performances in solving the real problem shows that all methods yielded the same result because the problem included jobs with similar processing times that were categorized based on certain features. GRASP is deemed superior as it strikes a balance between exploring new solutions and utilizing the best solution discovered thus far, thereby increasing the likelihood of uncovering the optimal solution.

PFSPs are common and can be found in real-world production environments often. Therefore, optimizing the PFSP impacts production efficiency and makes them an important area of research and development. The findings demonstrate the potential benefits of adopting this approach and can serve as a valuable reference for other companies considering similar solutions.

In conclusion, this study revealed that successful results could be obtained using SS, GA, and GRASP to solve PFSP. In future studies, the success of the methods can be retested by taking longer-term data from the company. Also, this problem can be tested by changing the parameters of the methods and using them in a hybrid way, or other metaheuristic algorithms can be developed for PFSP solutions in future studies.

REFERENCES

Allahverdi, A. (2003). The two and m-machine flow shop scheduling problem with bi-criteria of makespan and mean flow time. *European Journal of Operational Research*, 147: 373–396.

Arroyo, J.E.C. and de Souza Pereira, A. A. (2011). A GRASP heuristic for the multi-objective permutation flowshop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 55(5): 741-753.

Babaei, M., Mohammadi, M., Ghomi, S. M. T. F. and Sobhanallahi, M. A. (2012). Two parameter-tuned metaheuristic algorithms for the multi-level lot sizing and scheduling problem. *International Journal of Industrial Engineering Computations*, 3(5): 751–766.

Bautista, J., Cano, A., Companys, R., & Ribas, I. (2012). Solving the FmI blockI Cmax problem using bounded dynamic programming. *Engineering Applications of Artificial Intelligence*, *25*(6), 1235-1245.

Ben-Daya, M. and Al-Fawzan, M. (1998). A tabu search approach for the flow shop scheduling problem. *European Journal of Operational Research*, l09, 88-95.

Borovska, P. (2006, June). Solving the travelling salesman problem in parallel by genetic algorithm on multicomputer cluster. In *International Conference on Computer Systems and Technologies-CompSysTech* (Vol. 6, No. 2.11).

Bozejko, W. and Wodecki, M. (2008). Parallel Scatter Search Algorithm for the Flow Shop Sequencing Problem. Wyrzykowski, R., Dongarra, J., Karczewski K. and Wasniewski, J. (Eds.). *Parallel Processing and Applied Mathematics* (pp.180-188). Springer-Verlag Berlin Heidelberg.

Campbell, H. G., Dudek, R. A. and Smith, M. L. (1970). A Heuristic Algorithm for the n job, m Machine Sequencing Problem. *Management Science*, 16(10): B630-B637.

Çiçekli, U. G. and Bozkurt, S. (2015). *Permütasyon Akış Tipi Çizelgeleme Probleminin Dağınık Arama İle Optimizasyonu*. 15. Üretim Araştırmaları Sempozyumu, pp: 443-452, İzmir, 14-16 Ekim, 2015.

Çiçekli, U. G. and Kaymaz, Y. (2016). A Genetic Algorithm For The Allocation of Dangerous Goods Containers In A Storage Yard For Freight Villages and Dry Ports. *Gazi University Journal of Faculty of Economics and Administrative Sciences*, 18(1): 264-282.

Feo, T. and Resende, M.G.C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6: 109–134.

Festa, P. and Resende, M. G. (2002). GRASP: An annotated bibliography. In *Essays and surveys in metaheuristics* (pp. 325-367). Springer, Boston, MA.

Goldberg D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, USA.

Gupta, J. (1972). Heuristic algorithms for multistage flowshop scheduling problem. *AIIE Transactions*, 4(1):11–18.

Haq, A..N., Saravanan, M., Vivekraj A. R. and Prasad T. (2007). A Scatter Search Approach for General Flowshop Scheduling Problem. *The International Journal of Advanced Manufacturing Technology*, 31(7-8): 731–736.

Huang, M. W., Hsieh, C. C. and Arora, J. S. (1997). A genetic algorithm for sequencing type problems in engineering design. *International Journal for Numerical Methods in Engineering*, *40*(17), 3105-3115.

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

Shahul Hamid Khan, B. S. H., Prabhaharan, G. and Asokan, P. (2007). A Grasp algorithm for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness. *International Journal of Computer Mathematics*, 84(12): 1731-1741.

Kocamaz, M. and Çiçekli, G. (2010). Paralel Makinaların Genetik Algoritma İle Çizelgelenmesinde Mutasyon Oranının Etkinliği [Efficiency of Mutation Rate for Parallel Machine Scheduling with Genetic Algorithm]. Ege Academic Review, 10(1): 199-210.

Kocamaz, M., Cicekli, U. G. and Soyuer, H. (2009, July). A developed encoding method for parallel machine scheduling with permutation genetic algorithm. In *European and Mediterranean Conference on Information Systems EMCIS*.

Laguna, M. and Marti, R. (2003). *Scatter search: methodology and implementations in C*. Springer Science & Business Media.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs*, Berlin: Springer.

Molina-Sánchez, L. and González-Neira, E. (2016). GRASP to minimize total weighted tardiness in a permutation flow shop environment. *International Journal of Industrial Engineering Computations*, *7*(1), 161-176.

Nawaz, M., Enscore Jr., E.E. and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA, The International Journal of Management Science,* 11(1): 91–95.

Nowicki, E. and Smutnicki, C. (2006). Some aspects of scatter search in the flow-shop problem. European Journal of Operational Research, 169 (2): 654–666.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & operations research*, *22*(1), 5-13.

Resende, M. G. and Ribeiro, C. C. (2014). GRASP: Greedy randomized adaptive search procedures. In *Search methodologies* (pp. 287-312). Springer, Boston, MA.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research,* 165: 479–494.

Ruiz, R., Maroto, C. and Alcara, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research,* 165: 34–54.

Schaller, J. and Valente, J. M.S. (2013). A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness. *International Journal of Production Research*, 51(3): 772-779.

Shahsavari Pour, N., Tavakkoli-Moghaddam, R. and Asadi, H. (2013). Optimizing a multi-objectives flow shop scheduling problem by a novel genetic algorithm. *International Journal of Industrial Engineering Computations*, 4(3): 345–354.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278-285.

Vallada, E. and Ruiz, R. (2010). Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, 38(1-2): 57-67.

Widmer, M., Hertz, A. and Costa, D. (2008). *Metaheuristics and Scheduling, in Production Scheduling* (eds P. Lopez and F. Roubellat), ISTE, London, UK. doi: 10.1002/9780470611050.ch3

Yenisey, M. M., & Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, *45*, 119-135.

Zhang, L. and Wu, J. (2014). A PSO-Based Hybrid Metaheuristic for Permutation Flowshop Scheduling Problems. *Hindawi Publishing Corporation The Scientific World Journal*, 1-8.