An Application of Permutation Flowshop Scheduling Problem in Quality Control Processes

Chapter · January 2019

DOI: 10.1007/978-3-319-92267-6_68

Göksu Erseven
Yasar University

Gizem Akgün

Özgün Yücel
Yasar University

Adalet Öner
Yasar University

An Application of Permutation Flowshop Scheduling Problem in Quality Control Processes

Göksu Erseven, Gizem Akgün, Aslıhan Karakaş, Gözde Yarıkcan, Özgün Yücel, and Adalet Öner(✉)

Department of Industrial Engineering, Yaşar University, İzmir, Turkey
goksuersvn@hotmail.com, gizemakgunn1510@gmail.com, karakasaslihn@hotmail.com, gozdeyrkcn@hotmail.com, ozgun.ozturk@yasar.edu.tr, adalet.oner@yasar.edu.tr

Abstract. This study involves in a real world application of permutation flowshop scheduling problem (PFSP) in a local company that produces apparel and garments. Various test jobs arrive in the quality control department and a schedule has to be prepared considering the test stations and processing times. A job consists of a series of operations on certain stations. Each station is designed to perform a particular test operation. Therefore, there are jobs waiting to go through a series of quality control operations on various stations. Each job has different processing times at stations. Moreover, different jobs may have different processing times at a particular test station. However, the operation sequences (routes) of all jobs are the same through the stations. It is not allowed to change the order of the jobs on different stations. This problem falls into the realm of *permutation flow shop scheduling problem (PFSP*) in the literature. Various mathematical models are defined in literature for the optimal solution of the problem. One of them was chosen and used to solve small-sized problems. However, as the size of the problem increases, and since the problem is shown to be NP-hard for three or more machines, the solution time increases rapidly and hence it becomes significantly hard to solve the problem in polynomial time. Therefore, a combination of two heuristic methods is employed to solve the problem in reasonable time. First, NEH (Nawaz, Enscore, Ham) method is used to obtain a good initial feasible or a near optimal solution and then iterated local search method (ILS) is engaged to improve the solution. This solution procedure is implemented in a decision support tool in order to develop efficient schedules for quality control jobs in the company. The performance of the procedure is evaluated and verified by comparing the solutions with the optimal solutions of small sized problems. The tool can also be used for educational purposes since it is user friendly and has ability to present outcomes in detail with proper graphics.

Keywords: Permutation Flow Shop Scheduling Problem · Optimization Heuristics · Iterated Local Search · Decision support system

N. M. Durakbasa and M. G. Gencyilmaz (Eds.): ISPR 2018, *Proceedings of the International Symposium for Production Research 2018*, pp. 849–860, 2019.
https://doi.org/10.1007/978-3-319-92267-6_68

1 Introduction

A local company produces apparel and garments for men and women. The competitive attribute of the company is the quality of its products. It exports most of the products. In order to keep the competitive edge and to meet consumer safety standards and regulatory requirements of its destination markets, the company established and currently operates a sophisticated quality control facility equipped with state of art test stations. Among others, some of the quality tests performed in the facility are listed below.

- fiber identification
- test for banned azo colorants
- chemical testing
- dimensional stability: torque, stretch & recovery and shrinkage
- abrasion or piling
- colorfastness test
- flammability and burn test

The facility is busy to check and test the samples coming from different sources. The main source is the regular samples drawn from production lines. The other source is the samples of new designs. Each sample corresponds a job in the facility. A planning tool is desired in order to prepare efficient weekly schedules for the test jobs waiting in the system. In the facility, each job is processed through the same series of tests in a predefined order. That order is the same for all jobs. Each job has different processing times at different stations. Moreover, different jobs may have different processing times at a particular test station. The nature of the problem fits in the *permutation flow shop scheduling problem* in the literature.

Permutation flow shop scheduling problem (PFSP) has been extensively studied in the literature and has important applications in manufacturing and service systems. In the traditional permutation flow shop scheduling problems, n jobs are processed on m machines in the same order. The aim is to find the best sequence of jobs to be processed. The objective function is usually defined as minimizing the makespan. However, there are some other performance criteria employed in literature such as flow time, earliness, lateness, tardiness etc. [1]. Pinedo [2] defines the problem in detail and describe a scheme of classification for extensions and variants of PFSP. Wagner [3] presents initial methods and mathematical models for the solutions of job shop and permutation flowshop scheduling problems. Several other researchers provide mathematical models as in Baker [4], Stafford [5], Wilson [6] and Manne [7].

Rinnooy Kan [8] explains PFSP is a NP-hard problem for three or more machines. Therefore, many heuristics and meta-heuristics are proposed to solve the problem. Constructive heuristics and improvement heuristics are the two main categories of PFSP heuristics. The NEH algorithm is proposed by Nawaz et al. [9] is an example of constructive methods. There are other constructive algorithms presented by Palmer [10] and Campbell et al. [11]. According to Dong et al. [12] and Li et al. [13], NEH heuristic is considered as the best constructive heuristic for solving PFSP.

Meta-heuristic algorithms have also been studied to solve the problem. There are methods based on meta-heuristics such as genetic algorithms, simulated annealing [14], tabu search [15] and ant colony optimization [16]. Also, the iterated local search (ILS) [17] method is a good example for improvement heuristics. ILS is a simple but powerful metaheuristic that have mechanisms to run away from local minima/maxima. Those mechanisms include perturbation process, which is the best-known process to jump to a new restart position.

2 Problem Definition

The scheduling problem in garments and apparel company is a permutation flow shop scheduling problem as described above. The PFSP can be defined simply as follows: There are a set of n jobs and a set of m machines. Each job should pass through a set of m operations which must be done on different machines. All jobs have the same processing order of operations while passing through the machines. There are no precedence constraints among operations of different jobs. Operations cannot be interrupted and each machine can process only one operation at a time. The sequence changes between machines are not allowed. The objective is to find the best job sequence, which minimize the makespan, i.e. the maximum of the completion times of all operations.

The test operations in the company are organized to be done on 7 stations which correspond to the machines in the definition above. Similarly, the test samples correspond to jobs. The nature of the operations and the specifications of the machinery imposes pre-emptive mode of processing. The jobs are ready at the beginning of planning horizon.

There are some mathematical models reported in literature such as Baker [4], Stafford [5], Wilson [6] and Manne [7]. Those models have been investigated and the model proposed by Manne [7] has been chosen to implement for its simplicity. The details of that model are as follows.

$i \in \{1, \ldots, m\}$ stands for machine index

$j \in \{1, \ldots, n\}$ stands for job index.

Decision Variables:

$$D_{jj'} = \begin{cases} 1, & \text{if job j is scheduled before (not necessarily immediately before) job j}' \\ 0, & \text{otherwise} \end{cases}$$

C_{ij} = Completion time of job's j operation on machine i

C_{max} = Finishing time of the last operation on machine m (makespan)

Parameters:

p_{ij} = Processing time of job j on machine i

M = Large number

$$Z = \min C_{max} \tag{1}$$

$$\text{s.t.} \quad C_{1j} \geq p_{1j} \qquad j = 1, \ldots\ldots, n \tag{2}$$

$$C_{ij} - C_{i-1j} \geq p_{ij} \qquad i = 2, \ldots, m;\ j = 1, \ldots, n \tag{3}$$

$$C_{ij} - C_{ij'} + MD_{jj'} \geq p_{ij} \qquad i = 1, \ldots, m;\ j, j' = 1, \ldots., n, j < j' \tag{4}$$

$$C_{ij} - C_{ij'} + M\left(D_{jj'} - 1\right) \leq -p_{ij'} \qquad i = 1, .., m; j, j' = 1, \ldots, n, j < j' \tag{5}$$

$$C_{max} \geq C_{mj} \qquad j = 1, \ldots, n \tag{6}$$

$$C_{max} \geq 0 \tag{7}$$

$$C_{ij} \geq 0 \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \tag{8}$$

$$p_{ij} \geq 0 \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \tag{9}$$

The objective function given in (1) minimizes the makespan. Constraint (2) represents that the completion time of any job j is greater than or equal to the processing time for machine 1. Constraint (3) shows that the completion time difference of any job j between two successive machines $(i-1, i)$ is equal to or greater than the processing time in the i^{th} machine of the same job j. Constraints (4) and (5) include the machine availability. These constraints define the precedence relationship between the jobs j on any machine i. Here, big M corresponds a large number. These constraints insure that job j either precedes job j' or follows job j' in the sequence, but not both. Constraint (6) represents the makespan, which is equal or greater than the maximum completion time of all jobs on the last machine. Constraints (7), (8) and (9) are the non-negativity constraints.

The model has been implemented in order to see the optimal solutions for small sized problems. However, PFSP has been shown to be NP-hard for three or more machines. For this reason, as the size of the problem increases, the solution time increases rapidly and hence it becomes significantly hard to solve the problem in polynomial time. Therefore, heuristic solution methods have been investigated in order to have a way to solve the problem in reasonable time. NEH (Nawaz, Enscore, Ham) algorithm has been chosen as the constructive heuristics since it is simple, easy to implement and proven to be effective. Additionally, iterated local search (ILS) algorithm has been decided to be the improvement algorithm. Therefore, a combination of NEH and ILS algorithms are used to solve the problem. In this setting, NEH algorithm is employed to find a good feasible solution, and then ILS steps in to improve the solution provided by NEH algorithm.

3 NEH and ILS Algorithms

The NEH algorithm is proposed by Nawaz, Enscore and Ham [9] for permutation flow shop scheduling problems that minimizes the makespan. In the NEH algorithm, the jobs are first sorted in descending order depending on the sum of processing times on all machines. Then the first two jobs with highest sum of processing times on the machines are considered for partial scheduling. The best partial schedule of those two jobs (i.e., one that provides lower partial makespan) is determined. This partial sequence is fixed in a sense that the relative order of those two jobs will not change until the end of the procedure. In the next step, the job with the third highest sum of processing times is selected and three possible partial schedules are generated through placing the third chosen job at the beginning, in the middle, and at the end of the fixed partial sequence. These three partial schedules are examined and one that produces minimum partial makespan is chosen. This procedure is repeated until all jobs are fixed and the complete schedule is generated.

Iterated Local Search is a powerful meta-heuristic algorithm proposed by Stützle [17]. The main characteristics of this algorithm is to randomly leap in the determined solution area. With this algorithm, the local optimum solution is obtained. In order not to be confined to a single place, perturbations are made with splashing other places and new local optimum results are found.

The local search starts with some initial sequences and then continually tries to improve the existing sequence with local changes. If a better sequence is found in the neighborhood of the current directory, the current sequence replaces existing sequence and the local search continues. The simplest local search algorithm applies these steps repeatedly until a better sequence is found in the neighborhood and stops at the first local minimum encountered. The pseudo-code of ILS algorithm is shown below in Fig. 1.

π= NEH
π_{best}= π
While "(stopping criterion)" **do**
    π_1= Perturbation (π)
    π_2= Local Search (π_1)
    If f (π_2) < f (π) **then**
        $\pi = \pi_2$
        If f (π) < f (π_{best}) **then**
            $\pi_{best} = \pi$
End while
Return π_{best}
End ILS

Fig. 1. Pseudo-code of ILS

Perturbation is defined as neighborhood for the local search algorithm in the literature for flow type scheduling problems. First, jobs i and $i+1$ in two adjacent positions are interchanged (binary placement). Then, in the displacement phase, i^{th} and j^{th} job positions are reciprocally exchanged. Finally, the job in position i is removed and inserted in place of the job in position j (placement). The perturbation procedure is shown below (Fig. 2).

```
Do
pt1 = Select a random position of job
pt2 = Select a random position of job
   Loop While (pt1 =pt2)
   ward = Generate a random binary number (0, 1)
   If ward=1 Then
         Temp = p (pt2)
      Do
        p (pt2) = p (pt2 - 1)
        pt2 = pt2 - 1
      Loop While (pt2>pt1)
      p (pt2) = Temp
   Else
      Temp = p (pt1)
      Do
        p (pt1) = p (pt1 + 1)
        pt1 = pt1 + 1
      Loop While (pt1<pt2)
      p (pt1) = Temp
   End If
```

Fig. 2. Perturbation procedure

4 Decision Support System

Manne's mathematical model, NEH and ILS algorithms are implemented in the computer and embedded in a user friendly decision support tool. The interface of the tool is shown in Fig. 3. It provides some functions to help the user for setting and solving the problem quickly. One of the functions is designed for managing the input which includes the job list. A user may list, delete, add or edit the entries in the job list. Therefore, it ensures that the user has the updated version of the list. A part of this function is displayed in Fig. 4.

Once the job list is finalized, the decision tool processes input by incorporating a database, which holds the processing times of each job type, and hence prepares input data for PFSP. In order to solve the problem, there are three options. The first option is the optimal solution by mathematical model. However, the size of the problem is limited for this option since it takes a long time to solve the problem. A commercial

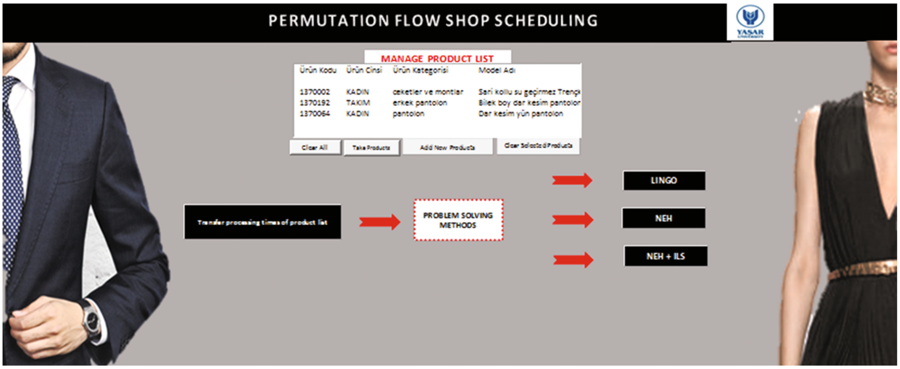

Fig. 3. DSS dashboard

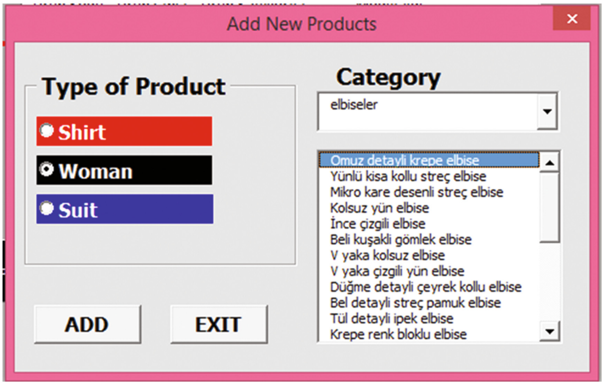

Fig. 4. Managing job list

solver sits behind to take part if mathematical model is desired to solve the problem within allowed size limits.

The second option is the solution by NEH algorithm only, and finally last option is the solution by the combination of NEH and ILS algorithms. The user may choose any option, and corresponding solution is shown in an Excel sheet formatted to present the outcomes in a form that is easy to understand.

The outcomes include the processing times of the jobs, starting and the completion times of the jobs on each station, the order in which the jobs will pass through the stations and the maximum completion time (Cmax). The detailed results may be printed in different formats as desired, and even sent in electronic format via e-mail. Additionally, the tool displays the Gantt chart of the solution. This feature is very useful to present outcomes in graphical format and hence it is easy to track the plan on different test stations. Sample Gantt charts are shown in Figs. 5 and 6.

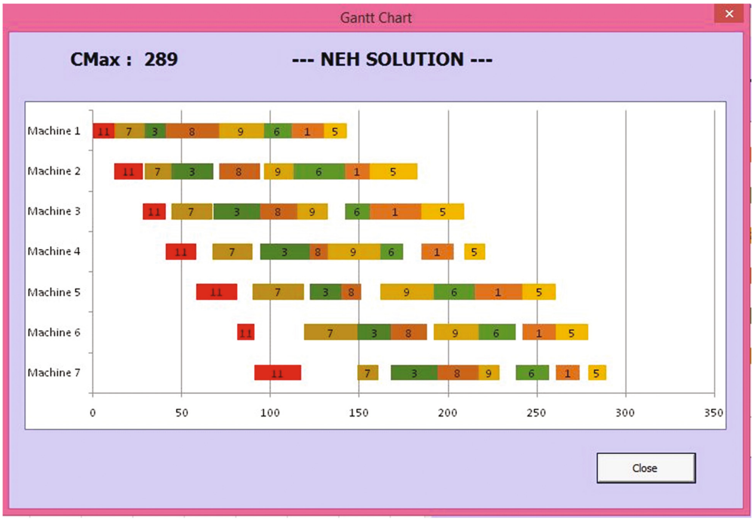

Fig. 5. Gantt chart for the NEH Solution

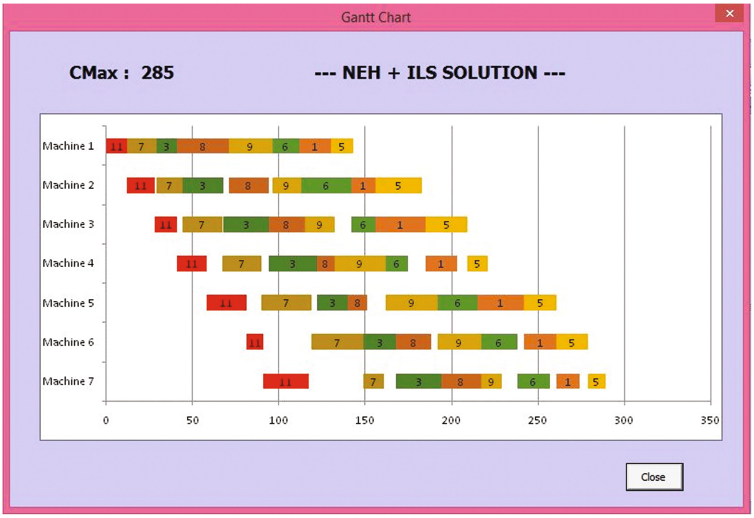

Fig. 6. Gantt chart for the NEH+ILS Solution

5 Computational Experience

This section includes verification of implementation of solution procedures and comparison of performances. All solution methods implemented in this tool has been verified by tracking numerous test problems. Numerical checks are supplemented with visual controls on Gantt charts. The mathematical model is implemented in Lingo 17.0. optimization software. In order to compare the performance of the solution procedures, different sets of test problems have been prepared. The inputs are generated randomly. The following tables show the performance comparisons of NEH, ILS and mathematical model with respect to their objective function values. Initial solution of the ILS algorithm is based on NEH algorithm. In the tables, *m* represents the number of machines and *n* represents the number of jobs. In Tables 1 and 2, the outcomes of seven different problem sets are presented. The problem sets are based on different number of jobs and machines. Each set includes 20 instances In Table 1, in all of the instances, except one, ILS algorithm reaches the optimal solution. In Table 2, ILS algorithm solution generates 0.07%, 0.56% and 0.31% average optimality gap in respective problems sets.

Table 1. Results of the Experiments

C_{MAX}	n = 6, m = 2			n = 6, m = 3			n = 8, m = 3			n = 8, m = 5		
Ins #	NEH	ILS	LINGO	NEH	ILS	LINGO	NEH	ILS	LINGO	NEH	ILS	LINGO
1	89	89	89	92	92	92	111	111	111	142	142	142
2	93	93	93	78	78	78	124	124	124	138	138	138
3	70	70	70	96	93	93	139	139	139	149	149	149
4	84	84	84	91	91	91	97	96	96	130	129	129
5	89	89	89	81	81	81	99	99	99	120	118	118
6	62	62	62	94	94	94	68	68	68	137	137	137
7	73	73	73	66	66	66	97	97	97	139	138	138
8	60	60	60	77	74	74	130	130	130	151	147	147
9	71	71	71	95	93	93	105	105	105	133	133	133
10	64	64	64	79	79	79	106	105	103	129	128	128
11	90	90	90	98	98	98	126	126	126	138	136	136
12	88	88	88	95	95	95	119	116	116	132	129	129
13	59	59	59	78	78	78	105	105	105	143	142	142
14	78	78	78	111	111	111	113	113	113	129	129	129
15	76	76	76	81	81	81	136	136	136	153	151	151
16	64	64	64	66	66	66	97	96	96	148	141	141
17	73	73	73	93	93	93	78	77	77	113	113	113
18	89	89	89	92	91	91	94	94	94	131	128	128
19	74	74	74	99	99	99	105	105	105	99	97	97
20	70	70	70	103	101	101	92	92	92	165	160	160

Table 2. Continuation of the Results of the Experiments

C_{MAX}	n = 10, m = 3			n = 10, m = 5			n = 10, m = 6		
Ins #	NEH	ILS	LINGO	NEH	ILS	LINGO	NEH	ILS	LINGO
1	145	145	145	159	152	150	161	159	159
2	139	135	133	189	189	184	152	151	151
3	135	132	132	150	148	148	161	159	159
4	135	132	132	147	142	142	170	169	169
5	120	116	116	145	145	141	171	169	169
6	146	146	146	142	139	139	174	170	170
7	116	116	116	159	159	159	184	177	170
8	136	131	131	155	155	153	196	188	188
9	123	123	123	180	170	169	164	162	162
10	136	136	136	152	149	149	182	179	179
11	135	135	135	154	154	152	170	169	169
12	154	154	154	164	161	160	171	163	163
13	123	123	123	172	172	172	202	197	197
14	116	116	116	146	145	145	158	157	157
15	147	147	147	136	129	129	187	183	183
16	141	141	141	146	145	145	170	163	163
17	143	143	143	179	179	179	194	186	186
18	118	118	118	169	169	169	176	175	171
19	118	117	117	153	146	145	168	168	168
20	126	126	126	171	169	169	158	158	158

In Table 3, relatively large instances are considered and optimal solution cannot obtained within a one-hour time limit. Hence, results of seven different problems set, each having 20 instances, are summarized in Table 3 for both NEH and ILS algorithms. Average gap between NEH and ILS is calculated for all problem sets and the smallest gap is obtained in first problem set (n = 12, m = 3) as 0.21% and the largest gap is obtained in problem set 5 (n = 15, m = 10) as 2.92%.

Table 3. NEH and ILS Comparisons

C_{MAX}	n = 12, m = 3		n = 12, m = 4		n = 12, m = 6		n = 12, m = 8		n = 15, m = 10		n = 20, m = 10		n = 30, m = 15	
Ins #	NEH	ILS	NEH	ILS	NEH	ILS	NEH	ILS	NEH	ILS	NEH	ILS	NEH	ILS
1	144	142	159	158	217	210	231	219	288	278	335	334	547	531
2	155	154	147	147	201	201	237	237	298	277	338	327	513	505
3	129	129	171	170	169	168	235	230	301	297	299	296	553	535
4	167	167	136	134	186	186	208	208	273	271	339	337	512	506
5	159	159	152	152	208	192	252	252	290	284	342	334	531	514
6	153	153	147	145	181	181	203	192	283	271	324	316	534	528
7	164	164	197	193	191	184	195	194	276	266	320	317	530	521
8	159	157	190	190	181	178	241	231	270	264	304	301	500	498
9	123	122	182	182	186	185	233	232	298	295	345	337	532	513
10	147	147	149	143	173	167	238	229	291	276	350	341	515	512
11	168	168	159	158	194	190	246	243	294	268	333	331	521	513
12	163	163	160	154	209	203	214	214	303	289	314	313	504	498
13	134	134	181	179	179	179	243	233	281	276	319	307	548	544
14	132	132	173	173	204	204	236	233	285	279	362	351	529	517
15	179	179	187	184	187	186	209	207	297	286	359	357	474	469
16	136	136	184	181	195	194	200	195	283	280	343	328	545	527
17	156	156	159	159	210	204	233	215	256	256	344	337	543	541
18	157	157	173	173	205	200	217	212	280	279	336	323	552	530
19	132	132	163	159	197	197	229	229	284	275	339	328	530	530
20	126	126	158	158	189	189	225	220	280	275	369	364	518	513

6 Conclusion

This study performs a PFSP in a quality control department of a textile company. As the problem is NP-hard, gathering optimal solutions by a solver (LINGO) is highly expensive. Therefore, NEH and ILS heuristic algorithms are implemented in order to minimize the makespan. Small and large problem sets are generated and small instances reach the optimal solution in a specified time limit. In those instances, NEH algorithm is used as an initial sequence to the ILS algorithm and maximum optimality gap is obtained as 2.76% with respect to ILS algorithm. Relatively large problem instances cannot reach the optimal solution in a given time limit due to the complexity of the problem. However, most of the cases, ILS improves the makespan with respect to NEH heuristic.

A flexible and user friendly tool has been developed for the company to enable the user to prepare quick and efficient schedules. It has been a useful tool for the company since they don't have such a tool before this study. It manages the input of the problem and present the solutions to the user through a user-friendly interface. The solution process and outcomes has been verified and validated. The tool has considerably decreased the amount of scheduling effort since schedules were prepared manually earlier.

The tool can also be used for educational purposes since it is user friendly and has ability to present outcomes in detail with proper graphics. It may be improved further in this field if animation effect can be added.

Acknowledgment. This work cannot be completed without the assistance of Anıl Tekye, Selin Gökkaya and Sinan Maramuroğlu. We are thankful for the contribution of them.

References

1. Chakraborty UK (2009) Computational intelligence in flow shop and job shop scheduling. Springer Science & Business Media, Berlin
2. Pinedo M (2002) Scheduling – theory, algorithms, and systems. Prentice-Hall, Upper Saddle River
3. Wagner HM (1959) An integer linear-programming model for machine scheduling. Nav Res Logist Q 6:131–140
4. Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York
5. Stafford EF (1988) On the development of a mixed-integer linear programming model for the flowshop sequencing problem. J Oper Res Soc 39:1163–1174
6. Wilson JM (1989) Alternative formulations of a flow-shop scheduling problem. J Oper Res Soc 40:395–399
7. Manne AS (1960) On the job-shop scheduling problem. Oper Res 8:219–223
8. Rinnooy Kan AHG (1976) Machine scheduling problems: classification, complexity, and computations. Nijhoff, The Hague
9. Nawaz M, Enscore EE Jr, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. OMEGA 11(1):91–95
10. Palmer DS (1965) Sequencing jobs through a multistage process in the minimum total time: a quick method of obtaining a near-optimum. Oper Res Q 16:101–107
11. Campbell HG, Dudek RA, Smith ML (1970) A heuristic algorithm for the n job, m machine sequencing problem. Manag Sci 16(10):B630–B637
12. Dong X, Huang H, Chen P (2008) An improved NEH-based heuristic for the permutation flowshop problem. Comput Oper Res 35:3962–3968
13. Li XP, Wang YX, Wu C (2004) Heuristic algorithms for large flowshop scheduling problems. In: Proceedings of the 5th world congress on intelligent control and automation, pp 2999–3003
14. Osman I, Potts C (1989) Simulated annealing for permutation flow shop scheduling. OMEGA 17(6):551–557
15. Grabowski J, Wodecki M (2004) A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion. Comput Oper Res 31(11):1891–1909
16. Rajendran C, Ziegler H (2004) Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. Eur J Oper Res 155(2):426–438
17. Stützle T (1998) Applying iterated local search to the permutation flowshop problem. Technical report, AIDA-98-04, Intellctics Group, Computer Science Department, Darmstad University of Technology, Darmstad, Germany