A Comparison of Constructive Heuristics with the Objective of Minimizing Makespan in the Flow-Shop Scheduling Problem

Pavol Semančo and Vladimír Modrák

Faculty of Manufacturing Technologies with seat in Prešov, TUKE, Bayerova 1, 08001 Prešov, Slovakia, pavol.semanco@tuke.sk, vladimir.modrak@tuke.sk

Abstract: We propose a constructive heuristic approach for the solution of the permutation flow-shop problem. The objective function of all algorithms is the minimization of the makespan. Our approach employs Johnson's rule to give a good initial solution for the improvement heuristic, also known as metaheuristics. The proposed heuristic algorithm, named MOD, is tested against four other heuristics that are well-known from the open literature, namely, NEH, Palmer's Slope Index, CDS and Gupta's algorithm. The computational experiment itself contains 120 benchmark problem data sets proposed by Taillard. We compare our results to the solutions represented by NEH outputs. The computational experiment shows that the proposed algorithm is a feasible alternative for practical application when solving n-job and m-machine in flow-shop scheduling problems to give relatively good solutions in a short-time interval.

Keywords: heuristic algorithm; NEH; Palmer's Slope Index; CDS; Gupta's algorithm; benchmark problem; flow shop

1 Introduction

Many manufacturing industries meet with the problem of how to effectively commit resources between varieties of possible orders in the current competitive environment. The searching for an optimal allocation of resources to performing a set of jobs within each work order is the main role of scheduling, which has become a necessity decision-making process in manufacturing. The main problems in scheduling of jobs in manufacturing are, according to Wight [24], "priorities" and "capacity". Hejazi and Saghafian [4] characterize the scheduling problem as an effort "to specify the order and timing of the processing of the jobs on machines, with an objective or objectives."

In this paper, we focus on an environment where all jobs have to follow the same route in the same order and where machines are assumed to be set up in a series,

which is also referred to as a flow shop. We consider general flow-shop scheduling with unlimited intermediate storage, where it is not allowed to sequence changes between machines. In this flow shop, referred to as permutation flow shop, the same job sequence of jobs is maintained throughout.

Although we limited our attention to only permutation schedules with constant setup times that are included in processing times and to availability of all jobs at zero time, these kinds of algorithms can be used to improve logistic chains of container transport as well [16].

The general flow-shop problem with a makespan (Cmax) objective can be denoted as an n/m/F/Cmax that involves n jobs where each requiring operations on m machines, in the same job sequence. The solution of such problem is represented by the optimal job sequence that produces the smallest makespan, assuming no preemption of jobs. The general flow-shop problem is also assumed as NP-hard for m>2.

We propose a constructive heuristic approach, based on application of Johnson's algorithm for the solution of the NP-hard flow-shop problem. Our approach uses a pair-splitting strategy to create a two-machine problem. We provide empirical results for Taillard's problem, instances demonstrating the efficacy of the approach in finding a good initial speed.

Common algorithms to solve NP-hard problems are heuristics giving solutions that do not necessarily have to be close to the optimum. However, they give good initial solutions in a reasonable time. Based on the literature, there are two well-known types of heuristics: constructive and improvement heuristics. The constructive heuristic starts without a schedule or job sequence and adds one job at a time. The most popular constructive heuristics are CDS [1] and NEH [11]. Improvement heuristics use as a initial position a schedule, mostly represented by the result of constructive heuristic, and they try to find a better "similar" schedule, referred to as improved solution. These iterative approaches, referred to as meta-heuristic approaches, are inherently local search techniques, such as, for example, tabu search (TS), simulated annealing (SA), genetic algorithms (GA), etc.

We test our approach on a dataset including 120 benchmark problems of Taillard [20]. We compare the results of four constructive heuristics, namely MOD, CDS, Gupta's algorithm and Palmer's slope index algorithm, with the well-known NEH algorithm set as a reference algorithm.

The next section covers a review of the relevant literature of the flow-shop scheduling heuristics. Section 3 analyzes the formal description of the MOD approach. In Section 4, we provide a discussion of computational experiment and results. Section 5 reports a summary of the paper and discusses possible future research ideas.

2 Literature Review

The scheduling literature provides a rich knowledge of the general flow-shop scheduling problem to get permutation schedules with minimal makespan. It can be stated that this is a very popular topic in scheduling circles.

Taylor [21] and Gantt [2], the inventor of well-known Gantt charts that are still accepted as important scheduling tools today, give the first scientific consideration to production scheduling. Pinedo [17] is a superior reference for all types of scheduling problems, including flow-shop environment together with tutorials, and scheduling systems. Vieira et al. [23] present a framework for rescheduling when differences between the predetermined schedule and its actual realization on the shop floor effect of disturbances in the performance of the system.

Production scheduling systems that emerged later were mostly connected to shop floor tracking systems and were dispatching rules to sequence the work [5]. Similar scheduling systems are today implemented in ERP systems that were performed in the early 1990s.

Modrak [10] discusses manufacturing execution systems (MES) with integrated scheduling systems in the role of a link interface between a business level and shop floor.

Heuristic solutions for the permutation flow-shop scheduling problem range from constructive heuristics, such as CDS and the NEH algorithm, to more complex approaches, known as meta-heuristics, namely branch and bound, tabu search, genetic algorithms and the ant colony algorithm.

Johnson [6] first presented an algorithm that can find the optimum sequencing for an n-job and 2-machine problem. The concept of a slope index as a measure to sequence jobs was firstly introduced by Page [14]. Later on, Palmer [15] adopted this idea and utilized the slope index to solve job sequencing for the m-machine flow-shop problem. Gupta [3] argued that the sequencing problem is a problem of sorting n items to minimize the makespan. He proposed alternative algorithm for calculating the slope index to schedule a sequence of jobs for more than two machines in a flow-shop scheduling problem.

Campbell et al. [1] proposed a simple heuristic extension of Johnson's algorithm to solve an m-machines flow shop problem. The extension is known in literature as the Campbell, Dudek, and Smith (CDS) heuristic.

Nawaz et al. [11] proposed the NEH algorithm, which is probably the most well-known constructive heuristic used in the general flow-shop scheduling problem. The basic idea is that a job with the largest processing time should have highest priority in the sequence. Results obtained by Kalczynski and Kamburowski [7] have also given proof that many meta-heuristic algorithms are not better than the simple NEH heuristic. The proof is also supported by famous "No Free Lunch"

(NFL) theorem, which points out that all algorithms equal to the randomly blind search if no problem information is known [25]. The solution quality greatly depends on the technique selection, which does not necessarily need to perform as well on other types of problem instances if it fits a specific type of problem instances.

The most emphasized names among the contributors of meta-heuristic approaches are as follows: Ogbu and Smith [13] with their simulated annealing approach; Nowicki and Smutnicki [12], who implemented tabu search to solve the flow-shop scheduling problem; And Reeves and Yamada [18], who applied the genetic algorithm for PFSP. The new accession to the family of meta-heuristic scheduling algorithms is a water-flow like algorithm [22]. The Hybrid algorithm, based on the genetic algorithm, was applied in order to find optimal makespan in an n-job and m-machine flow-shop production, see [19].

In this paper, we focus on using Palmer, Gupta, CDS and NEH heuristics against MOD approach. For details on these heuristics, see [1], [3], [8], [11] and [15].

3 Constructive Heuristic MOD

In this section we formally explain the steps of the constructive heuristic approach used to obtain a good initial solution. Further details of this heuristic are referred to in [9]. The general idea is that we adopt the Johnson's rule in the last step of our proposed algorithm to get the minimum makespan. We use the difference between the sums of processing times for each machine as a pair-splitting strategy to make two groups of the matrix of n-job and m-machine. We further explain our approach mathematically.

3.1 Notation

The following notations were used:

J	set of n jobs $\{1, 2, \ldots, n\}$
M	set of m machines $\{1, 2, \ldots, m\}$
M_p	set of two pseudo machines $\{1, 2\}$
G	set of 2 clusters $\{I, II\}$
k	number of k machines
l	number of m-k machines
I	cluster of k machines
II	cluster of m-k machines

p_{ij} processing time of *i*th job on *j*th machine, $i \in J$ and $j \in M$

P_j sum of processing time of *n* jobs on *j*th machine

ΣPg total sum of processing time of *n* jobs on machines in *g*th group, $g \in G$

it well-fitting iteration number

DIF_{it} difference between groups *I* and *II*

C_{max} makespan

C_j completion time

s splitting ratio

s_{max} max splitting ratio

3.2 MOD Algorithm

Step 1: Calculate the sum of processing time

$$P_j = \sum_{i=1}^{n} p_{ij} \forall i \in J,\ j \in M \tag{1}$$

Step 2: Compute the total sum of the processing time for each cluster

Calculate ΣP_I , ΣP_{II} of cluster *I* and *II* as follows:

$$\sum P_I = \sum_{j=1}^{k} P_j \forall k \in I,\ j \in M \tag{2}$$

$$\sum P_{II} = \sum_{j=m-k}^{m} P_j \forall k \in I,\ j \in M \tag{3}$$

Step 3: *Compute the splitting ratio and apply the pair-splitting strategy*

Compute the splitting ratio for this iteration given by:

$$s_k = \frac{\min(\Sigma P_I ; \Sigma P_{II})}{\max(\Sigma P_I ; \Sigma P_{II})} \tag{4}$$

Apply the pair-splitting strategy:

a. If s_k is the maximum ratio so far, save the current *k* as well-fitting iteration (*it*) and the ratio as the maximum ratio (s_{max}).

b. If $k = m$ then go to *Step 5.*

c. If $s_k = 1$, go to *Step 5.*

Step 4: Next iteration

Increment k by one and go back to *Step 2.*

Step 5: Compute the completion time for each cluster and create two pseudo machines

Calculate the completion time C_j of ith job for both clusters according to following formulas ($k = it$):

a. Cluster *I*:

$$C_j = k \cdot p_{1j} + (k-1) \cdot p_{2j} + \ldots + p_k \tag{5}$$

b. Cluster *II*:

$$C_j = l \cdot p_{1j} + (l-1) \cdot p_{2j} + \ldots + p_l \tag{6}$$

Tabulate these values into two rows to get two pseudo machines (M_{p1}, M_{p2}).

Step 6: Apply Johnson's rule on the two pseudo machines

Apply Johnson's rule on the two pseudo machines of n jobs to get the job sequence.

Step 7: Display the solution

The C_{max} of particular job sequence from *Step 6* is the solution.

3.3 Pair-Splitting Strategy and Parameters

In Step 3 of Section 3.2, there are two splitting parameters, namely the splitting ratio (s) and well-fitting iteration number. We explain each of these parameters next.

3.3.1 Splitting Ratio

The splitting ratio is one of the parameters that control the degree of similarity of two created clusters. The pair of clusters with the highest rate is used for further computation. The splitting ratio ranges from 0 to 1, where 1 indicates the same size of two clusters and vice versa.

3.3.2 Well-fitting Iteration

We also build a parameter to backtrack the best pair of clusters created from the n-job and m-machine mechanism matrix. The well-fitting iteration parameter also indicates the number of machines for the cluster *I*.

4 Computational Experiments

We ran our experiment with objective of minimizing the makespan on Taillard's benchmark problem datasets, which has 120 instances, 10 each of one particular size. Taillard's datasets range from 20 to 500 jobs and 5 to 20 machines. The outputs of the NEH algorithm were used as reference solutions for comparison purposes.

4.1 Platform and Parameters

We coded the MOD, NEH, CDS, Palmer's Slope Index and Gupta's algorithms in PHP script, running on a PC with a 3.06 GHz Intel Core and 2GB of RAM. All PHP-coded algorithms have a user-friendly interface with the possibility to select whether to run each heuristic individually or altogether. It has also an option to draw a Gantt chart with a legend.

4.2 Performance Measures

We used a relative percent deviation (RPD) and an average relative percent deviation (ARPD) as performance measures for comparing the solutions of each algorithm to the reference solutions.

The relative percent deviation and average percentage relative deviation is given by:

$$RPD_i = \frac{HS_i - RS_i}{RS_i} \cdot 100\% \tag{7}$$

$$ARPD = \frac{1}{I} \cdot \sum_{i=1}^{I} RPD_i \tag{8}$$

where: I number of problem instances,

HS_i heuristic solution of problem instance i,

RS_i reference solution of problem instance i,

RPD_i percentage relative deviation of problem instance i.

4.3 Results

In the computational experiment, we use the problem instances described earlier. The summary results for Taillard's 120 instances are shown in tables 1 to 4. Each of the summary tables displays the results for MOD, CDS, Gupta's algorithm,

Palmer's Slope Index and the NEH alone. The computational experiment takes the performance indicators of the algorithms to be the solution quality (C_{max}) and runtime (CPU). Tables 1 to 3 show the computational results of makespans and RPDs for each algorithm and for each problem instance.

Table 1

Makespans and RPDs for Taillard's 20-job and 50-job benchmark-problem datasets

Problem instance	NEH (Reference makespan)	MOD		CDS		Gupta		Palmer	
		C_{max}	RPD	C_{max}	RPD	C_{max}	RPD	C_{max}	RPD
20x5									
1	1299	1322	1.8	1436	10.5	1400	7.8	1384	6.5
2	1365	1433	5.0	1424	4.3	1380	1.1	1439	5.4
3	1132	1136	0.4	1255	10.9	1247	10.2	1162	2.7
4	1329	1475	11.0	1485	11.7	1554	16.9	1420	6.8
5	1305	1355	3.8	1367	4.8	1370	5.0	1360	4.2
6	1251	1299	3.8	1387	10.9	1333	6.6	1344	7.4
7	1251	1366	9.2	1403	12.2	1390	11.1	1400	11.9
8	1215	1312	8.0	1395	14.8	1410	16.0	1290	6.2
9	1284	1371	6.8	1360	5.9	1444	12.5	1426	11.1
10	1127	1235	9.6	1196	6.1	1194	5.9	1229	9.1
20x10									
1	1681	1789	6.4	1833	9.0	2027	20.6	1790	6.5
2	1766	1802	2.0	2021	14.4	1960	11.0	1948	10.3
3	1562	1621	3.8	1819	16.5	1780	14.0	1729	10.7
4	1416	1575	11.2	1700	20.1	1730	22.2	1585	11.9
5	1502	1714	14.1	1781	18.6	1878	25.0	1648	9.7
6	1456	1607	10.4	1875	28.8	1650	13.3	1527	4.9
7	1531	1650	7.8	1826	19.3	1761	15.0	1735	13.3
8	1626	1799	10.6	2056	26.4	2084	28.2	1763	8.4
9	1639	1731	5.6	1831	11.7	1837	12.1	1836	12.0
10	1656	1917	15.8	2010	21.4	2137	29.0	1898	14.6
20x20									
1	2443	2787	14.1	2808	14.9	2821	15.5	2818	15.3
2	2134	2331	9.2	2564	20.1	2586	21.2	2331	9.2
3	2414	2598	7.6	2977	23.3	2900	20.1	2678	10.9
4	2257	2541	12.6	2603	15.3	2670	18.3	2629	16.5
5	2370	2615	10.3	2733	15.3	2868	21.0	2704	14.1
6	2349	2439	3.8	2707	15.2	2722	15.9	2572	9.5
7	2383	2465	3.4	2684	12.6	2796	17.3	2456	3.1
8	2249	2467	9.7	2523	12.2	2612	16.1	2435	8.3
9	2306	2550	10.6	2617	13.5	2701	17.1	2754	19.4
10	2257	2557	13.3	2649	17.4	2690	19.2	2633	16.7
50x5									
1	2729	2839	4.03	2883	5.64	2820	3.33	2774	1.65
2	2882	3152	9.37	3032	5.20	2975	3.23	3014	4.58
3	2650	2850	7.55	3010	13.58	3071	15.89	2777	4.79
4	2782	2925	5.14	3179	14.27	3102	11.50	2860	2.80
5	2868	2882	0.49	3188	11.16	3114	8.58	2963	3.31

Table 2
Makespans and RPDs for Taillard's 50-job and 100-job benchmark-problem datasets

Problem instance	NEH (Reference makespan)	MOD		CDS		Gupta		Palmer	
		C_{max}	RPD	C_{max}	RPD	C_{max}	RPD	C_{max}	RPD
50x5									
6	2835	2959	4.37	3175	11.99	3104	9.49	3090	8.99
7	2806	3021	7.66	3005	7.09	3109	10.80	2845	1.39
8	2700	2827	4.70	3189	18.11	3091	14.48	2826	4.67
9	2606	2783	6.79	3171	21.68	3211	23.22	2733	4.87
10	2801	2827	0.93	3224	15.10	3092	10.39	2915	4.07
50x10									
1	3175	3468	9.23	3671	15.62	3672	15.65	3461	9.01
2	3073	3174	3.29	3645	18.61	3577	16.40	3313	7.81
3	2994	3191	6.58	3677	22.81	3670	22.58	3335	11.39
4	3218	3417	6.18	3707	15.20	3645	13.27	3511	9.11
5	3186	3417	7.25	3664	15.00	3499	9.82	3427	7.56
6	3148	3340	6.10	3584	13.85	3559	13.06	3318	5.40
7	3277	3539	8.00	3784	15.47	3723	13.61	3457	5.49
8	3170	3407	7.48	3744	18.11	3746	18.17	3382	6.69
9	3025	3422	13.12	3518	16.30	3561	17.72	3414	12.86
10	3267	3370	3.15	3913	19.77	3699	13.22	3404	4.19
50x20									
1	4006	4347	8.51	4759	18.80	4645	15.95	4272	6.64
2	3958	4370	10.41	4414	11.52	4354	10.01	4303	8.72
3	3866	4265	10.32	4469	15.60	4485	16.01	4210	8.90
4	3953	4360	10.30	4793	21.25	4773	20.74	4233	7.08
5	3872	4218	8.94	4642	19.89	4649	20.07	4376	13.02
6	3861	4320	11.89	4505	16.68	4714	22.09	4312	11.68
7	3927	4138	5.37	4758	21.16	4665	18.79	4306	9.65
8	3914	4295	9.73	4609	17.76	4577	16.94	4318	10.32
9	3970	4277	7.73	4465	12.47	4543	14.43	4547	14.53
10	4036	4222	4.61	4556	12.88	4488	11.20	4197	3.99
100x5									
1	5514	5929	7.53	5602	1.60	5765	4.55	5749	4.26
2	5284	5436	2.88	5669	7.29	5697	7.82	5316	0.61
3	5222	5323	1.93	5638	7.97	5531	5.92	5325	1.97
4	5023	5310	5.71	5287	5.26	5269	4.90	5049	0.52
5	5261	5424	3.10	5584	6.14	5535	5.21	5317	1.06
6	5154	5278	2.41	5203	0.95	5200	0.89	5274	2.33
7	5282	5530	4.70	5557	5.21	5434	2.88	5376	1.78
8	5140	5230	1.75	5509	7.18	5504	7.08	5263	2.39
9	5489	5538	0.89	5821	6.05	5901	7.51	5606	2.13
10	5336	5593	4.82	5740	7.57	5670	6.26	5427	1.71
100x10									
1	5897	6208	5.27	6749	14.45	6549	11.06	6161	4.48
2	5466	5745	5.10	6285	14.98	6238	14.12	5889	7.74
3	5747	6043	5.15	6648	15.68	6359	10.65	6119	6.47
4	5924	6368	7.49	6848	15.60	6908	16.61	6329	6.84
5	5672	6025	6.22	6399	12.82	6499	14.58	6070	7.02
6	5395	5852	8.47	6136	13.73	6154	14.07	5870	8.80
7	5717	6359	11.23	6417	12.24	6535	14.31	6442	12.68
8	5752	6300	9.53	6513	13.23	6425	11.70	6168	7.23
9	6016	6304	4.79	6356	5.65	6386	6.15	6081	1.08
10	5937	6287	5.90	6835	15.13	6816	14.81	6259	5.42

Table 3
Makespans and RPDs for Taillard's 100-job, 200-job and 500-job benchmark-problem datasets

Problem instance	NEH (Reference makespan)	MOD		CDS		Gupta		Palmer	
		C_{max}	RPD	C_{max}	RPD	C_{max}	RPD	C_{max}	RPD
100x20									
1	6520	7092	8.77	7584	16.32	7668	17.61	7075	8.51
2	6550	7194	9.83	7615	16.26	7600	16.03	7058	7.76
3	6621	7350	11.01	7526	13.67	7628	15.21	7181	8.46
4	6589	7226	9.67	7909	20.03	7802	18.41	7039	6.83
5	6697	7057	5.38	7681	14.69	7628	13.90	7259	8.39
6	6813	7234	6.18	7582	11.29	7832	14.96	7109	4.34
7	6578	7156	8.79	8125	23.52	7892	19.98	7279	10.66
8	6791	7425	9.34	7902	16.36	8098	19.25	7567	11.43
9	6679	7017	5.06	7668	14.81	7687	15.09	7271	8.86
10	6680	7267	8.79	7947	18.97	7557	13.13	7305	9.36
200x10									
1	10949	11631	6.23	12151	10.98	12220	11.61	11443	4.51
2	10677	11236	5.24	12088	13.22	12170	13.98	10986	2.89
3	11080	11575	4.47	12378	11.71	11948	7.83	11336	2.31
4	11057	11397	3.07	11730	6.09	11676	5.60	11265	1.88
5	10615	11202	5.53	11634	9.60	11604	9.32	11125	4.80
6	10495	11438	8.99	11854	12.95	11592	10.45	10865	3.53
7	10950	11554	5.52	12436	13.57	12055	10.09	11333	3.50
8	10834	11361	4.86	11801	8.93	12088	11.57	11275	4.07
9	10565	11230	6.29	12197	15.45	12189	15.37	11184	5.86
10	10808	11436	5.81	11758	8.79	11893	10.04	11355	5.06
200x20									
1	11638	12750	9.55	13446	15.54	13724	17.92	13042	12.06
2	11678	12494	6.99	13129	12.43	13132	12.45	12813	9.72
3	11724	12799	9.17	13578	15.81	13651	16.44	12846	9.57
4	11796	12734	7.95	13297	12.72	13608	15.36	13061	10.72
5	11670	12559	7.62	13004	11.43	13132	12.53	12827	9.91
6	11805	12491	5.81	13583	15.06	13233	12.10	12381	4.88
7	11876	12511	5.35	13110	10.39	13175	10.94	12584	5.96
8	11824	12561	6.23	13799	16.70	13929	17.80	12824	8.46
9	11801	12886	9.19	13289	12.61	13407	13.61	12523	6.12
10	11890	12862	8.17	13709	15.30	13720	15.39	12615	6.10
500x20									
1	26774	28551	6.64	30650	14.48	29851	11.49	28246	5.50
2	27215	29031	6.67	30838	13.31	29804	9.51	29439	8.17
3	26941	28432	5.53	30532	13.33	29960	11.21	28073	4.20
4	26928	28342	5.25	30208	12.18	30372	12.79	28058	4.20
5	26928	28286	5.04	29917	11.10	29540	9.70	27768	3.12
6	27047	28428	5.11	29866	10.42	29868	10.43	28516	5.43
7	26820	28116	4.83	30428	13.45	29955	11.69	27878	3.94
8	27230	28293	3.90	30073	10.44	30021	10.25	28294	3.91
9	26541	27892	5.09	29120	9.72	30065	13.28	27745	4.54
10	27103	28979	6.92	30232	11.54	30498	12.53	28313	4.46
ARPD			**6.88**		**13.54**		**13.36**		**7.10**

The final line of Table 3 gives the overall average RPD values over all problem instances.

The solutions of developed algorithm and those of CDS, Gupta's algorithm, and Palmer's Slope Index algorithm are compared with the NEH optimal solutions for problems with a size up to 20 machines and 500 jobs.

From the results we can also make the following observations. Overall, the MOD heuristic performed better than any of tested algorithms with the exception of the NEH algorithm that was used as reference heuristic for this study. MOD's average RPD for all 120 Taillard's problems came at 6.88%. The cost of computing time was insignificant, in contrast to other three tested algorithms.

For Taillard's 20-job problems, i.e., 20x5, 20x10 and 20x20 size problems, MOD found the closest match to the reference solutions for 19 of the 30 problems. The average RPD of the MOD approach for the 20-job problems came at 8.06%. Thus, MOD performed very well on the 20-job Taillard's problems. For 50-job problems, i.e., 50x5, 50x10 and 50x20 size problems, MOD's average relative percentage deviation was 6.97%, which is the smallest ARPD of all four algorithms. For the 100-job problems, MOD's varied by the overall size of the problem. The 100x5 problems were solved to within an average RPD of 3.57%, while the 100x20 problems came at an average RPD of 8.28%.

Instead of displaying the times for each problem individually, we grouped the average computational times for each size of the problem. The average computational times (CPU) are summarized for each size of the problem and depicted in Figure 1. The CPU times, as can be seen from the graph, vary by the size of the problem. For example, MOD took between 153 and 157 milliseconds for 500-job problems. CDS took from 86 to 90 milliseconds and NEH from 617248 to 640114 milliseconds for 500x20 problems.

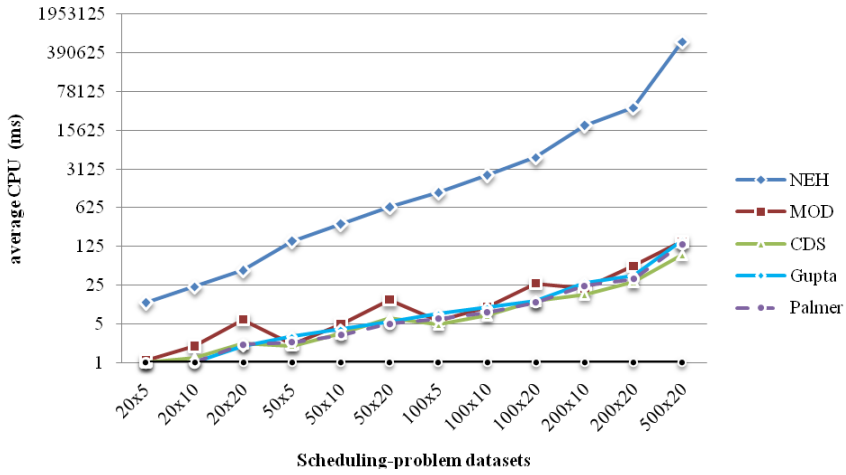

Figure 1
Average CPU times for each group of the problems

Conclusions

In the presented study, a constructive heuristic based on Johnson's rule is presented for the sequencing problem with sequence-dependent jobs, which is a quite common problem in many industries. The approach uses pair-splitting strategy and tries to find the minimal makespan. Based on the tested problems involving multiple jobs and machines, the proposed approach proved that is capable of good results. The proposed algorithm gave the best performance of all four approaches. The average RPD from the reference algorithm was 6.88% for all Taillard's problems.

The MOD approach was used to give a better solution than three other heuristics, namely Palmer, CDS and Gupta. For all three heuristics, the MOD algorithm showed significant improvements and compared well with the best-known NEH heuristic. Empirical testing on 120 benchmark problems drawn from Taillard produced some very good results.

We thus make an important contribution by proposing a new constructive heuristic for solving the permutation flow-shop scheduling problem with the objective of minimizing the makespan. The MOD algorithm finds near-optimal solutions for many benchmark problems in a reasonable time.

Future research could address this approach to more difficult flow-shop problems involving sequence-dependent setup times. Different objective functions can also be tested. Larger problems could be attempted with this approach. Future research can further try to find better pair-splitting strategies.

Acknowledgement

This work has been supported by the Grant Agency of the Ministry of Education of the Slovak Republic and Slovak Academy of Sciences (VEGA project No. 1/4153/07 "Development and application of heuristics methods and genetic algorithms for scheduling and sequencing of production flow lines").

References

[1] Campbell, H. G. et al.: A Heuristic Algorithm for The n-job, m-machine Scheduling Problem, Management Science 16, 1970, pp. 630-637

[2] Gantt, H. L.: Organizing for Work, New York: Harcourt, Brace, and Howe, 1919

[3] Gupta, J. N. D.: A Functional Heuristic Algorithm for the Flow Shop Scheduling Problem. Operational Research Quarterly 22, 1971, pp. 39-47

[4] Hejazi, S. R., and Saghafian, S.: Flowshop Scheduling Problems with Makespan Criterion: A Review, International Journal of Production Research, 43(14), 2005, pp. 2895-2929

[5] Herrmann, J. W.: Handbook of Production Scheduling, New York: Springer, 2006

[6] Johnson, S. M.: Optimal Two- and Three-Stage Production Schedules with Setup Times Included, Naval Research Logistics Quarterly 1, 1954, pp. 61-68

[7] Kalczynski, P. J. et al.: On the NEH Heuristic for Minimising the Makespan in Permutation Flowshops, The International Journal of Management Science, 35 (1), 2007, pp. 53-60

[8] Malindzak, D.: Models and Simulation in Logistics, Acta Montanistica Slovaca, 15 (1), 2010, pp. 1-3

[9] Modrak, V. et al.: Flow Shop Scheduling Algorithm to Minimize Completion Time for n-jobs m-machines Problem, Tehnicki Vjesnik, 17(3), 2010, pp. 273-278.

[10] Modrak, V. et al.: Mapping Development of MES Functionalities, Proceedings of 6th International Conference on Informatics in Control, Automation and Robotics, 2009, pp. 244-247

[11] Nawaz, M. et al.: A Heuristic Algorithm for the m-machine, n-job Flow Shop Sequencing Problem, International Journal of Management Science 11, 1983, pp. 91-95

[12] Nowicki, E. et al.: A Fast Tabu Search Algorithm for the Permutation Flow-Shop Problem, Eur. J. Oper. Res. 91, 1996, pp. 160-175

[13] Ogbu, F. A. et al.: The Application of the Simulated Annealing Algorithm to the Solution of the n/m/c Subscript Max Flowshop Problem, Computers Ops Res. 17, 1990, pp. 243-253

[14] Page, E. S.: An Approach to Scheduling of Jobs on the Machines, J. Royal Stat. Soc., 23, 1961, pp. 484-492

[15] Palmer, D. S.: Sequencing Jobs through a Multi-Stage Process in the Minimum Total Time - a Quick Method of Obtaining a Near Optimum, Operational Research Quarterly 16, 1965, pp. 101-107

[16] Pap, E. et al.: Application of Pseudo-Analysis in the Synchronization of Container Terminal Equipment Operation, Acta Polytechnica Hungarica, 8(6), 2011, pp. 5-21

[17] Pinedo, M.: Scheduling: Theory, Algorithms and Systems, New Jersey: Springer, 2008

[18] Reeves, C. R. et al.: Genetic Algorithms, Path Relinking, and Flow Shop Problem, Evolutionary Computation 6, 1998, pp. 45-60

[19] Semanco, P. et al.: Hybrid GA-based Improvement Heuristic with Makespan Criterion for Flow-Shop Scheduling Problems, Communications in Computer and Information Science, 220, 2011, pp. 11-18

[20] Taillard, E.: Benchmarks for Basic Scheduling Problems, European Journal of Operational Research 64, 1993, pp. 278-285

[21] Taylor, F. W.: Principles of Scientific Management, New York and London: Harper & brothers, 1911

[22] Tran, T. H. et al.: A Water-Flow Algorithm for Flexible Flow Shop Scheduling with Intermediate Buffers, Journal of Scheduling, 14 (5), 2010, pp. 483-500

[23] Vieira, G. E. et al.: Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods, Journal of Scheduling, 6(1), 2003, pp. 35-58

[24] Wight, O. W.: Production and Inventory Management in the Computer Age, New York: Van Nostrand Reinhold Company, Inc. 1984

[25] Wolpert, D. H. et al.: No Free Lunch Theorems for Optimization, Proceedings of IEEE Trans. Evol. Comput, 1 (1), 1997, pp. 67-82