BETTER HEURISTICS FOR M-MACHINE NO-WAIT FLOWSHOP WITH TOTAL COMPLETION TIME CRITERION

Ali Allahverdi[1] **and Tariq Aldowaisan**[1]

1: Associate professor, Department of Industrial and Management Systems Engineering, Kuwait University

E-mail: allahverdi@kuc01.kuniv.edu.kw

ABSTRACT

This paper presents several new heuristics for the m-machine no-wait flowshop with total completion time as the criterion. The performance of the proposed heuristics is compared with that of three existing heuristics including a recently developed Genetic Algorithm. Computational experience demonstrates the superiority of the proposed heuristics with respect to error performance. For example, for number of jobs 400 and number of machines 25, the suggested proposed heuristics PH1(p) and PH3(p) yield an average percentage relative error of 0.006% and 0.257%. This is compared with an average percentage relative error of 2.764% for the best performing existing heuristic.

Keywords: *Scheduling, Flowshop, no-wait, total completion time, heuristics.*

الملخص

تعرض هذه الدراسة عدة خوارزميات جديدة تهدف إلى تقليل الوقت الشامل للإنجاز في ورشة انسيابية مكونة من عـدة مكائن ودونما توقف . ولقد تم إجراء مقارنة بين الخوارزميات المقترحة مع خوارزميات ثلاث إحداها حديثة تعتمد على الخوارزم الجيني ، وجائت النتائج لصالح الخوارزميات المقترحة فيما يتعلق بنسبة الخطأ . فعلى سبيل المثال ، عنـدما يكون عدد المكائن ٢٥ وعدد المهام المطلوب إنجازها ٤٠٠ ، كان متوسط نسـبة الخطـأ النسـبي فـي اثنـين مـن الخوارزميات المقترحة هو ٠,٠٠٦% و ٠,٢٥٧% أما في أفضل الخوارزميات الموجودة فلقد جائت النتيجة ٢,٧٦٤%.

1. INTRODUCTION

In an m-machine flowshop problem, there are n jobs to be scheduled on m machines where each job consists of m operations and each operation requires a different machine and all jobs are processed in the same order of the machines. Extensive research has been conducted for many variants of the regular flowshop problem with the assumption that there is an infinite buffer space between the machines. Even though such an assumption is valid for some applications there are numerous situations in which discontinuous processing is not allowed. Such flowshops are known as no-wait.

A no-wait flowshop problem occurs when the operations of a job have to be processed continuously from start to end without interruptions either on or between machines. This means, when necessary, the start of a job on a given machine is delayed in order that the operation's completion coincides with the start of the next operation on the subsequent machine.

There are several industries where the no-wait flowshop problem applies. Examples include the metal, plastic, chemical and food industries. For instance, in the case of rolling of steel, the heated metal must continuously go through a sequence of operations before it is cooled in order to prevent defects in the composition of the material. Also in the food processing industry, the canning operation must immediately follow the cooking operation to ensure freshness. Additional applications can be found in advanced manufacturing environments, such as just-in-time and flexible manufacturing systems.

The no-wait flowshop problem has attracted the attention of many researchers. [Hall and Sriskandarajah, 1996] give in a survey paper a detailed presentation of the applications and research on this problem. Considering that flowtime or completion time of a job is the same when the job is ready for processing at time zero and that total or mean completion time are equivalent criteria, some of the works on the no-wait problem with the objective of minimizing any of these criteria include [Adiri and Pohoryles, 1982], [Rajendran and Chaudhuri, 1990], [van der Veen and van Dal, 1991], and [Chen et al., 1996].

[Adiri and Pohoryles, 1982] and [Van der Veen and Van Dal, 1991] consider special cases of m-machine no-wait flowshop problem for the total and mean completion time criteria. [Adiri and Pohoryles, 1982] prove some properties of the optimal schedules for a two-machine and provide theorems that are the basis for polynomial bounded algorithms for the m-machine with an increasing or decreasing series of dominating machines. [Van der Veen and Van Dal, 1991] show that the problem is solvable when the objective function is restricted to semi-ordered processing time matrices.

[Aldowaisan and Allahverdi, 1998], [Aldowaisan, 2000], and [Allahverdi and Aldowaisan, 2000] also consider the no-wait flowshop problem but with separate setup times. [Aldowaisan and Allahverdi ,1998] address the problem with a total flowtime performance measure for a two-machine. They develop optimal solutions for special cases, establish a local dominance relation, and provide a heuristic solution for the generic case. [Aldowaisan, 2000] considers the same problem. He provides additional dominance relations and proposes a new heuristic. [Allahverdi and Aldowaisan, 2000] address the three-machine problem with the same objective. They provide optimal solutions for certain cases, a dominance relation for the general case, and develop and evaluate five heuristic algorithms.

[Rajendran and Chaudhuri, 1990] and [Chen et al., 1996] address the m-machine generic no-wait problem. [Rajendran and Chaudhuri, 1990] give two heuristic algorithms and show that they are superior to other existing heuristics. [Chen et al., 1996] later develop a genetic algorithm and compare it with the heuristics of [Rajendran and Chaudhuri, 1990].

In this paper, we address the generic m-machine no-wait flowshop problem with the objective of minimizing total completion time. We propose six new heuristics and compare with the heuristics of [Rajendran and Chaudhuri, 1990] and [Chen et al., 1996].

2. HEURISTICS

We propose four heuristics for the m-machine no-wait flowshop problem and compare the proposed heuristics with the best three existing heuristics.

2.1. Existing Heuristics

Let $t_{i,j}$ denote the processing time of the job in position i on machine j. Also let $d_{i,j}$ denote the minimum delay between the start of the job in position i and that of the job in position j on the first machine due to the no-wait constraint. Let

$$A_i = \sum_{j=1}^{m} (m-1+j)t_{i,j} \quad \text{and} \quad B_i = \sum_{j=1}^{m} t_{i,j}$$

2.1.1. Heuristic RC1 (Rajendran and Chaudhuri, 1990)

Step 1: Form an initial sequence by arranging the jobs in ascending order of the value of A_i. If any tie exists choose the job having the least value of B_i.

Step 2: Set k=1, select the first job in the initial sequence and insert it in the first position of the partial sequence π.

Step 3: Update k=k+1. Select the kth job from the initial sequence and insert it in all r possible positions of the current solution of π where r is the integer satisfying $k/2 \leq r \leq k$. Among r sequences, select the one with the minimum value for the following expression

$$\sum_{i=2}^{k} (k+1-i)d_{i-1,i}$$

and make it current π.

Step 4: If k=n, stop, the sequence π is the heuristic solution of RC1, else go to Step 3.

2.1.2. Heuristic RC2 (Rajendran and Chaudhuri, 1990)

RC2 follows the same steps as RC1 except that it uses B_i as the ordering criterion for the initial sequence and A_i for tie breaking.

2.1.3. CNA Genetic Algorithm (Chen et al., 1996)

Genetic Algorithm (GA) is a probabilistic technique that imitates the evolution process. GA which was first developed by Holland (1975) has found many applications in different disciplines including scheduling, e.g., Nagar et al. (1996), Chen et al. (1996), Onwubolu and Mutingi (1999). For our problem, Chen et al. (1996) provides a GA with the following steps.

Step 1: Let the population size and the number of generations be 95 and 60, respectively. Determine the initial population, S(0) as follows. Half of the members in the initial population is generated randomly, and the other half is generated with the first one using Rajendran and Chaudhuri's (1990) method, the second one using the Danninbring (1997) method, and the following m-1 members generated using the CDS method (1970) (m is the number of machines). The remaining members of the second half are generated by randomly choosing a member and randomly swapping two of its jobs generate.

Step 2: For each member $s_i(t)$ in population S(t), compute the fitness value $f(s_i(t))$ as follows. First calculate the total flowtime for each member in the population. Second, calculate the fitness value for each member, which is equal to the difference between the maximum total flow time in the population and the total flow time of the member.

Step 3: For each member $s_i(t)$ in population S(t), compute the selection probability as $P(s_i(t)) = f(s_i(t))/\Sigma f(s_i(t))$.

Step 4: Based on the selection probability choose a pair of members (parents) for reproduction.

Step 5: Apply the genetic operators of crossover and mutation to the parents. Form a new population S(t+1) using the offspring of the parents. Go to Step 6 if the new population is equal to the population size. Otherwise go to Step 4.

Step 6: If the current generation t+1 is equal to 60 or the number of members in the population with the lowest total flow time is 60% of the population stop. Otherwise go to Step 2.

In step 5 of the algorithm, Goldberg's (1989) PMX operator is used for crossover with a crossover rate of 0.725. As for mutation operation, two positions in a member are randomly picked and swapped to generate a new member with a mutation rate of 0.009. These rates are determined based on empirical investigation. Other aspects of the CNA genetic algorithm are to include the best member in the current population in the next generation and to replace all members in the population with new ones except the best one. In developing the initial population, one of the members is generated by Rajendran and Chaudhuri's (1990). Since Rajendran and Chaudhuri's (1990) has two methods, we selected the one with the better performance.

Top

2.2. Proposed Heuristics

The proposed heuristics use the result of the following algorithm as an initial sequence.

2.2.1. Initial Sequence Algorithm (ISA)

Step 1: Set k=2, π_1={all jobs}, $\pi_2=\phi$

Step 2: Choose job i such that

$$\sum_{j=1}^{m} t_{i,j} \leq \sum_{j=1}^{m} t_{r,j} \text{ for all r in } \pi_1.$$

Remove job i from π_1 and place it in the first position of π_2.

Step 3: If k=n, stop, the sequence π_2 is ISA, else calculate the Total Completion Time of the jobs in positions 1, 2, …, k ($TCT_{1,k}$) for each job $i \in \pi_1$ after inserting it in position k of π_2 and assign the job with the minimum $TCT_{1,k}$ in position k in π_2 and remove it from π_1

Let k=k+1.

Step 4: Go to Step 3.

2.2.2. Proposed Heuristics 1 and 2 (PH1 and PH2)

The following steps describe the first two proposed heuristics where the only difference between the two is using different insertion methods in Step 3. While the first proposed heuristic (PH1) uses the Nawaz et al. (1983) insertion method, the second proposed heuristic (PH2) uses the Rajendran and Ziegler (1997) insertion method.

Step 1: Develop the initial sequence π_0 using the ISA. Let T_0 be the objective function value of the sequence π_0

Step 2: Set $T_b= T_0$, $\pi_b=\pi_0$, r=1

Step 3: Apply the Nawaz et al., 1983 (Rajendran and Ziegler, 1997) insertion method to the sequence π_{r-1} to obtain π_r and calculate T_r

Step 4: If $T_r< T_b$, set $T_b=T_r$ and $\pi_b=\pi_r$

Step 5: Update r=r+1. If r>10 go to Step 6, otherwise go to Step 3

Step 6: The PH1 (PH2) solution sequence is π_b with objective function value T_b.

2.2.3. Proposed Heuristics 3 and 4 (PH3 and PH4)

The following steps describe the second two proposed heuristics where the only difference between the two is using different insertion methods in Step 3. While the third proposed heuristic (PH3) uses the Nawaz et al. (1983) insertion method, the fourth proposed heuristic (PH4) uses the Rajendran and Ziegler (1997) insertion method.

Step 1: Develop the initial sequence π_0 using the ISA. Let T_0 be the objective function value of the sequence π_0

Step 2: Set $T_b= T_0$, $\pi_b=\pi_0$, r=1, k=0

Step 3: Apply the Nawaz et al., 1983 (Rajendran and Ziegler, 1997) insertion method to the sequence π_{r-1} to obtain π_r and calculate T_r

Step 4: If $T_r< T_b$, set $T_b=T_r$ and $\pi_b=\pi_r$

Step 5: If $T_r\geq T_b$, k=k+1

Step 6: Update r=r+1. If r>10 or k=2 go to Step 7, otherwise go to Step 3

Step 7: The PH3 (PH4) solution sequence is π_b with objective function value T_b.

The only difference between PH1 (PH2) and PH3 (PH4) is the stoppage criterion. While the former heuristics PH1 and PH2 terminate after 10 replications, the latter heuristics PH3 and PH4 terminate either after 10 replications or when two consecutive replicates yield a solution worse than the best solution obtained so far (i.e. k=2).

For all of the four proposed heuristics, a further improvement can be achieved by applying a pair-wise exchange procedure after the heuristic solution is obtained. The sequence obtained after the pair-wise exchange is applied to the proposed heuristic i (PHi) is denoted by PHi(p).

3. COMPARISON

In this section, we compare the performance of the heuristic algorithms using the FORTRAN language on a SUN SPARC Station 20. The processing times on each machine were randomly generated from a discrete uniform distribution, U(1,10) and U(1,100). Most researchers have used this distribution in their experimentation, e.g., Rajendran and Chaudhuri (1990).

The experiments are performed for the number of jobs of 50, 100, 200, 300, and 400, and the number of machines of 5, 10, 15, 20, and 25. The heuristics' solutions are compared with the best solution. The number of replicates is 30. We compare the performance of the heuristics using average percentage relative error (Avg), standard deviation (Std), and the number of best solutions (Cnt). The percentage relative error is defined as 100* (Heuristic – Best Solution)/Best Solution.

For the proposed heuristics PH1, PH2, PH3, and PH4 a statistic V is calculated to represent the average number of replicates to activate the stoppage criterion. Computational results revealed that PH4, which uses the efficient k=2 stoppage criterion, performs as good as PH2 in terms of error. Therefore, we decided not to compare PH2 with the other heuristics for the rest of the analysis.

Tables 1 and 2 provide the computational results of the existing and proposed heuristics, respectively. The superiority of the proposed heuristics over all three existing heuristics in terms of error is evident from Tables 1 and 2. The tables also show that PHi(p) performs better than PHi for i = 1, 3, and 4; and that PH1(p) provides the best error performance amongst all heuristics.

The CPU time of all heuristics (proposed and existing) is negligible. Even for the extreme case of 400 jobs, the CPU time of all heuristics is less than one minute. Therefore, we recommend PH1(p) based on its error performance.

4. CONCLUSIONS

This paper presents several proposed heuristics for the n-job m-machine no-wait flowshop problem with the objective of minimizing total completion time. The proposed heuristics differ in three aspects; firstly, in the choice between two stoppage criteria, secondly, in the choice between two insertion methods, and finally, in whether or not to apply a pair-wise exchange procedure.

The proposed heuristics are compared with three existing heuristics, two by Rajendran and Chaudhuri (1990) and a Genetic Algorithm by Chen et al. (1996). Computational experience for large number of jobs and for processing time distributions of U(1,10) and U(1,100) show that all of the proposed heuristics perform better than the existing ones in terms of error performance. Among the proposed ones, PH1(p) is recommended.

ACKNOWLEDGMENT

This research was supported by Kuwait University Research Administration project number EM-124.

Top

REFERENCES

1. Adiri, I. and Pohoryles, D., 1982, "Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times," *Naval Research Logistics Quarterly*, 29(3), pp 495-504.
2. Aldowiasan, T., 2000, "A new heuristic and dominance relations for no-wait flowshops with setups," *Computers & Operations Research*, 28(6), pp 563-584.
3. Aldowiasan, T., and Allahverdi, A., 1998, "Total flowtime in no-wait flowshops with separated setup times," *Computers & Operations Research,* 25(9), pp 757-765.
4. Allahverdi, A., Aldowaisan, T. 2000, "No-wait and separate setup three-machine flowshop with total completion time criterion," *International Transactions in Operational Research,* 7(3), pp 245-264.
5. Chen, C.L., Neppalli, R.V., and Aljaber, N., 1996, "Genetic Algorithms Applied to the Continuous Flow Shop Problem," *Computers & Industrial Engineering*, 30(4), pp 919-929.
6. Goldberg, D.E., 1989, Genetic algorithms in search, optimization, and machine learning. Addisson Wesley, Reading, M.A.
7. Hall, N.G., and Sriskandarajah, C., 1996, "A survey of machine scheduling problems with blocking and no-wait in process," *Operations Research*, 44(3), pp 510-525.
8. Holland, J.H., 1975, Adaptation in nature and artificial systems. University of Michigan Press, USA.
9. Nagar, A., Heragu, S.S., and Haddock, J., 1996, "A combined branch and bound and genetic algorithm based approach for a flowshop scheduling problem," *Annals of Operations Research,* 63, pp 397-414.
10. Nawaz, M., Enscore, E., and Ham, I., 1983, "A heuristic algorithm for the m-machine, n-job flowshop sequencing problem," *OMEGA The International Journal of Management Sciences,* 11, pp 91-95.
11. Onwubolu, G.C. and Mutingi, M., 1999, "Genetic algorithm for minimizing tardiness in flowshop scheduling," *Production Planning and Control*, 10(5), pp 462-471.
12. Rajendran, C., and Chaudhuri, D., 1990, "Heuristic Algorithms for Continuous Flow-Shop Problem," *Naval Research Logistics*, 37, pp 695-705.
13. Rajendran, C., and Ziegler, H., 1997, “An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs”, *European Journal of Operational Research,* 103(1), pp 129-138.
14. Van der Veen, J.A.A. and Van Dal, R., 1991, "Solvable cases of the no-wait flowshop scheduling problem," *Journal of the Operational Research Society*, 42(11), pp 971-980.

Top

Table 1: Performance of the existing heuristics

		CNA (Genetic)			RC1			RC2		
n	m	Avg.	Std.	Cnt	Avg.	Std.	Cnt	Avg.	Std.	Cnt
50	5	2.635	1.381	1	4.532	1.521	0	2.739	1.451	1
	10	1.624	1.097	0	2.932	1.504	0	2.016	1.348	0
	15	1.936	1.265	3	3.272	1.740	0	2.414	1.452	0
	20	2.040	1.122	0	2.782	1.468	0	2.698	1.454	0
	25	2.323	1.338	1	3.576	1.771	0	2.980	1.353	0
100	5	3.484	1.236	0	5.366	1.649	0	3.496	1.234	0
	10	2.404	1.106	0	3.853	1.247	0	2.606	1.267	0
	15	2.827	0.979	0	3.903	1.423	0	3.072	0.996	0
	20	2.930	1.266	0	4.294	1.374	0	3.147	1.310	0
	25	3.008	1.001	0	3.941	1.198	0	3.380	1.129	0
200	5	3.593	0.803	0	6.484	0.956	0	3.596	0.807	0
	10	2.627	0.803	0	4.361	0.899	0	2.635	0.809	0
	15	2.442	0.848	0	3.667	0.802	0	2.455	0.847	0
	20	2.806	0.819	0	3.819	0.803	0	2.843	0.845	0
	25	3.054	0.791	0	3.888	0.804	0	3.161	0.869	0
300	5	3.673	0.615	0	6.995	0.740	0	3.678	0.619	0
	10	2.474	0.601	0	4.300	0.705	0	2.475	0.601	0
	15	2.492	0.553	0	3.914	0.477	0	2.493	0.552	0
	20	2.717	0.540	0	3.698	0.737	0	2.734	0.554	0
	25	2.841	0.668	0	3.880	0.775	0	2.862	0.696	0
400	5	3.829	0.645	0	7.065	0.556	0	3.832	0.647	0
	10	2.323	0.456	0	4.163	0.605	0	2.325	0.457	0
	15	2.266	0.509	0	3.635	0.580	0	2.273	0.519	0
	20	2.554	0.531	0	3.788	0.677	0	2.585	0.546	0
	25	2.764	0.704	0	3.635	0.771	0	2.814	0.747	0

Table 2: Performance of the proposed heuristics

		PH1				PH1 (p)			PH3				PH3 (p)			PH4				PH4 (p)		
n	m	Avg.	Std.	Cnt	V	Avg.	Std.	Cnt	Avg.	Std.	Cnt	V	Avg.	Std.	Cnt	Avg.	Std.	Cnt	V	Avg.	Std.	Cnt
50	5	1.034	0.755	1	4.6	0.407	0.587	16	1.11	0.8	1	5.6	0.469	0.612	13	1.721	1.248	0	4.9	0.973	1.177	13
	10	0.25	0.282	6	3.9	0.108	0.244	22	0.325	0.305	6	4.9	0.147	0.258	18	1.202	1.426	4	5	1.003	1.286	8
	15	0.188	0.264	6	4.8	0.053	0.183	24	0.301	0.414	5	5.6	0.142	0.322	20	1.857	1.505	2	4.9	1.589	1.357	3
	20	0.261	0.342	6	4.2	0.09	0.236	23	0.357	0.432	5	4.8	0.199	0.363	20	1.455	1.155	5	5.2	1.331	1.084	6
	25	0.323	0.351	4	4.3	0.078	0.17	22	0.442	0.368	2	4.9	0.183	0.294	18	1.572	1.404	3	4.9	1.445	1.332	5
100	5	1.127	0.733	0	6.9	0.358	0.632	17	1.486	0.884	0	6	0.601	0.78	14	1.55	0.96	0	6.2	0.661	0.834	10
	10	0.32	0.266	1	5.8	0.103	0.217	21	0.557	0.568	1	5.5	0.295	0.582	19	1.465	0.79	0	6.2	1.152	0.793	6
	15	0.26	0.231	1	6.9	0.011	0.053	28	0.498	0.399	0	5.8	0.251	0.435	16	1.944	1.138	0	6.2	1.797	1.11	1
	20	0.149	0.133	3	7.1	0.012	0.045	28	0.629	0.784	2	5.8	0.409	0.686	17	2.184	0.99	0	6	1.988	1.018	0
	25	0.169	0.15	2	6.8	0.021	0.076	27	0.451	0.363	0	5.4	0.267	0.324	13	2.222	1.036	0	6	2.116	1.023	2
200	5	1.563	0.58	0	7.5	0.383	0.528	13	1.928	0.667	0	6.1	0.571	0.531	7	1.683	0.617	0	6.9	0.359	0.529	16
	10	0.3	0.19	0	8.4	0.036	0.102	24	0.555	0.439	0	6.9	0.209	0.353	17	1.499	0.864	0	7.7	1.113	0.869	3
	15	0.251	0.195	0	7.6	0.032	0.129	28	0.575	0.365	0	5.4	0.294	0.33	12	1.972	0.814	0	7.3	1.754	0.793	1
	20	0.166	0.117	0	7.4	0	0	30	0.585	0.576	0	5.5	0.416	0.547	13	1.98	0.831	0	7.6	1.855	0.802	0
	25	0.217	0.15	0	7.6	0.003	0.015	29	0.557	0.526	0	6.2	0.334	0.5	15	2.288	0.84	0	7.6	2.129	0.847	0
300	5	1.631	0.409	0	7.6	0.215	0.32	14	1.949	0.508	0	6.2	0.344	0.409	10	1.796	0.349	0	7.8	0.222	0.26	13
	10	0.343	0.184	0	8.5	0.039	0.149	25	0.667	0.364	0	6.2	0.293	0.33	12	1.571	0.694	0	7.7	1.173	0.675	2
	15	0.212	0.119	0	8.4	0	0	30	0.614	0.469	0	6.3	0.366	0.473	9	2.137	0.599	0	8	1.934	0.583	0
	20	0.172	0.069	0	8.2	0.003	0.015	28	0.398	0.371	0	7	0.197	0.318	18	2.186	0.654	0	7.7	2.023	0.635	0
	25	0.212	0.121	0	8	0.002	0.008	28	0.572	0.464	0	5.4	0.335	0.43	11	2.133	0.902	0	8.1	1.999	0.896	0
400	5	2.051	0.432	0	7	0.248	0.357	15	2.289	0.458	0	5.4	0.389	0.392	9	2.168	0.411	0	7.4	0.322	0.419	12
	10	0.385	0.133	0	8.4	0.027	0.064	24	0.675	0.318	0	5.8	0.253	0.308	12	1.247	0.673	0	7.9	0.786	0.64	1
	15	0.209	0.093	0	7.9	0.005	0.028	29	0.515	0.314	0	5.7	0.268	0.294	8	1.934	0.537	0	8.7	1.717	0.539	0
	20	0.191	0.093	0	6.9	0.003	0.016	29	0.558	0.443	0	5.2	0.321	0.41	10	2.158	0.618	0	8.7	2.004	0.605	0
	25	0.225	0.099	0	7.5	0.006	0.027	28	0.481	0.33	0	5.9	0.257	0.307	14	2.196	0.793	0	8.8	2.037	0.776	0

Top