African Journal of Business Management Vol. 6(17), pp. 5744-5749, 2 May, 2012
Available online at http://www.academicjournals.org/AJBM
DOI: 10.5897/AJBM11.2232
ISSN 1993-8233 (c)2012 Academic Journals

Full Length Research Paper

Proposing a clonal selection algorithm for hybrid flow shop

Mohammad Akhshabi[1]*, Mostafa Akhshabi[2] and Javad Khalatbari[3]

[1]Department of Industrial Engineering, Tonekabon Branch, Islamic Azad University, Tonekabon, Iran.
[2]Department of Computer Engineering, Tonekabon Branch, Islamic Azad University, Tonekabon, Iran.
[3]Department of Management, Tonekabon Branch, Islamic Azad University, Tonekabon, Iran.

Accepted 9 February, 2012

In this paper we propose a clonal selection algorithm (PCSA) to solve a hybrid flow shop scheduling (HFS) problem considering the minimization of the sum of the total earliness and tardiness penalties. In the view of its non-deterministic polynomial-time hard nature, so we propose the clonal selection algorithm to deal with this problem. The performance of our algorithm is tested by numerical experiments on a large number of randomly generated problems. By comparison with solutions, performance obtained by NEH heuristi (Nawaz et al., 1983) and the HC heuristi (Ho and Chang, 1991) is presented. The results show that the proposed approach performs well for this problem.

Key words: Hybrid flow shop, clonal selection algorithm, total earliness and tardiness penalties.

INTRODUCTION

The situation where an operation at a certain stage requires more than one machine (‘‘multiprocessor task’’) is addressed among others in Oguz et al. (2011) and Ying and Lin (2009). Some authors (Sawik, 2002; Wardono and Fathi, 2004, 2009) study the case of limited buffer capacities between production stages which lead to a ‘‘blocking scheduling problem’’ where a completed job remains on a machine and blocks it until a downstream machine becomes available. Contributions on hybrid flow shop (HSF) scheduling under no-wait constraints can be found for example Grabowski and Pempera (2000) and Wang et al. (2005). We point out here that the case of limited-wait constraints has only received limited attention: noticeable exceptions can be found in Yang and Chern (1995), Su (2003) and Akkerman et al. (2007). Finally, HFS with positive set up and/or removal times is studied (Low, 2006). However, as presented in Ruiz et al. (2008), even if there are several articles present in realistic extensions of the HSF scheduling, very few papers consider several complicating features jointly. Moreover, it can be seen from the detailed survey address in Ruiz and Vazquez-Rodriguez (2010), there seems to be no previous attempt in the operations of literature research to solve the variant of the HFS problem studied in the present paper, that is, the HFS problem with multiprocessor tasks, zero buffer capacity, limited waiting time between consecutive operations of a job and positive setup and removal times. There is a wide variety of solution approaches for the HFS problem. The authors of Ruiz and Vazquez-Rodriguez (2010) address a classification into three broad classes: exact methods, heuristics and metaheuristics. Exact solution approaches for the HFS problem mostly rely on problem-specific branch and bound algorithms where nodes correspond to partial schedules and lower bounds are computed by exploiting pacific properties of the HFS problem. But, as it can be seen in some surveys addressed in Ruiz and Vazquez-Rodriguez (2010), and Kis and Pesch (2010), research in this area focus on simplified diversions of the problem by considering either problems with a restricted number of processing stages (typically 2) or problems close to the standard form of the HFS. Moreover, existing branch and bound algorithms appear to be limited to situations where the objective is to minimize make span or mean flow time. In this study, we presented a HFS problem involving a total number of processing steps, several complicating job constraints and more complex

*Corresponding author. E-mail: m.akhshabi@toniau.ac.ir.

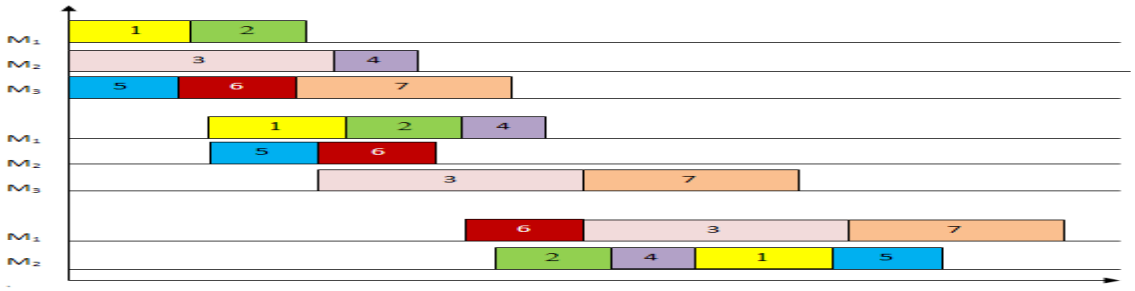

Figure 1. Gantt chart.

criterion, namely total weighted tardiness.

Problem formulation

The manufacturing environment of the HSF is considered as an extension of the classical flow shop. In fact, it presents a multistage production process with the property that a set of n jobs needs to be processed at all the stages in the same order; starting at stage 1 and finishing in stage S. Each stage consist of a given number m_i $(m_i \geq 1)$ of identical parallel machines available from time zero, and denoted $m_i = (m_{i1}, m_{i2}, \ldots, m_{im_i})$. So each job j needs several operations $(o_{1j}, o_{2j}, \ldots, o_{sj})$, where o_{ij} has to be processed by one machine out of a set of given machines at the i_{th} stage during an uninterrupted p_{ij} time unit (the preemption is not allowed) and can start only after the completion of the (i_1) previous operations. The starting time and the completion time of an operation are denoted by t_{ij} and c_{ij}, respectively. In this article, we assume that:

1) Setting up times of machines and moving times between operations are negligible.
2) Machines are independent of each other.
3) Jobs are independent of each other.
4) At a given time, a machine can only execute one operation.

Solving HFS scheduling problem consists of assigning operations to machines in each stage (routing problem) and sequencing the operations assigned to the same machine. The objective is to organize the execution of the n jobs on the machines in order to minimize an objective function. Defining c_j as the completion time and $d_j$as the due date of j ob j, we propose to develop schedules for the HFS problem that complete each customer order (which can be represented by a job j) at or near its due date d_j. In this scenario, both the early and tardy completion of jobs would be penalized. Earliness and tardiness penalties are the cost per unit of time a job is completed either earlier or later than requested by the customer. For a trivial solution, some jobs will have to be finished early and some other will have to be finished late (Finke et al., 2007). Consequently, it is suitable to provide schedules that minimize the earliness and tardiness penalties. These penalties could be different for jobs based on their priority importance. The minimization of these costs can be translated into the scheduling objective of minimizing a weighted sum of job earliness and tardiness (ET), which is given by Equation 1.

$$ET=\sum_{j=1}^{n}(w_{ej} \times E_j + w_{tj} \times T_j) \qquad (1)$$

Where E_j is the max {0, ($d_j\_c_j$)} is the earliness of job j; T_j the max {($c_j\_ d_j$), 0} is the tardiness of job j; w_{ej} is the penalty weight per unit of time when job j is produced early; w_{tj} is the penalty weight per unit of time when job j is produced late.

To illustrate the problem, let us consider a three-stage HFS scheduling problem where a set of n=7 jobs have to be processed through three stages. Then number of machines in each stage is, respectively, m_1=3, m_2=2and m_3=4. A feasible schedule is shown in the Gantt chart (Figure 1).

According to the processing order of the solution, we can determine the completion time c_j of each job j, and then calculate their earliness E_j and tardinessT_j. For example if, E_j=T_j=0 for all the jobs except jobs 7 and 5 where T_7=10 and E_5=5. Then, the objective function value of the proposed schedule which is given by Equation 2.

$$ET=\sum_{j=1}^{n}(w_{ej} \times E_j + w_{tj} \times T_j)=2\times 5+5\times 10 =60 \qquad (2)$$

According to the earliness and tardiness penalty weight, the schedule of a job j is preferably inserted into the set of early jobs if $w_{ej} < w_{tj}$, otherwise, it would preferably inserted into the set of tardy jobs. The relative importance of the earliness and tardiness penalties has been usually related to the tardiness factor (Almeida and Centeno, 1998).

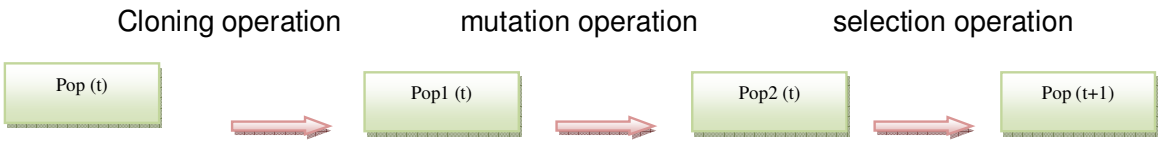

Figure 2. Basic state transfer flowchart of PCSA.

But, it is also important to focus on minimizing the earliness because satisfying a client's demand earlier than its due date may cause UN wanted inventory or product deterioration. Thus, we need to minimize the total earliness tardiness of scheduled jobs (M'Hallah, 2007). The problem is reduced to the special case of minimizing the total un-weighted ET, where $w_{ej} = w_{tj} = 1$ for all the jobs j which is given by Equation 3.

$$ET=\sum_{j=1}^{n}(E_j + T_j) \quad (3)$$

So, this study tackles the HFS scheduling problem to minimize the total earliness/tardiness of scheduled jobs.

Proposing clonal selection algorithm

PCSA contains three operations, which are cloning, mutation, and selection. A set of antibodies are selected, which have the best and the highest affinity with antigen from the antibodies. The selected antibodies are cloned in proportion to their affinity with the antigen (rank based). All the copies are mutated. The mutation degree is inversely rated to their parent's affinity. All the copies are added to the antibodies. A set of antibodies from all antibodies that have the best and the highest affinity with the antigen are re-selected. Figure 2 presents the basic state transfer flowchart of PCSA. Since clonal selection algorithm has the advantage of being simple to implement and easy to understand, it has shown considerable success in solving a variety of optimization problem. Clonal selection algorithm also has been reported in scheduling area. For example, Yang et al. (2008) proposed a clonal selection-based memetic algorithm for solving job shop scheduling problems. Kumar et al. (2006) extended the artificial immune system approach by proposing a new methodology termed as psycho-clonal algorithm to solve m-machine no-wait flow shop problem.

Encoding

Clonal are corresponding to the solutions of HFS. In our coding each Clonal consists of two parts of strings, A-string and B-string. A-string defines the routing policy of the problem, and B-string defines the sequence of the operations on each machine. The elements of the strings respectively describe a concrete allocation of operations to each machine and the sequence of operations on each machine.

Assuming that the total number of operations is l, and the i^{th} operation can be processed by a machine set $m_i = (m_{i1}, m_{i2}, \ldots, m_{im_i})$. The length of A-string is l and it can be denoted i^{th} as $g_1, g_2, \ldots, g_i$. The i^{th} element g_i is an integer between 1 an in and n_i it means that the i^{th} operation is assigned to $g_i{}^{th}$ the machine m_{ig_i} in m_i.

B-string has a length of l too. It is consisted of a sequence of job numbers in which job number j occurs o_j Times. o_j is the number of total operations that the j^{th} job comprises. When decoding a particle, B-string is converted to a sequence of operations at first. Then each operation is assigned to a processing machine according to A-string. At last each operation in the sequence will be scheduled in its earliest permitted time. In this way, each particle is corresponding to a feasible HFS schedule.

Initialization

The initial population of antibodies is randomly selected based on uniform distribution probability for all variables to cover the entire search space uniformly.

Clonal proliferation

The fittest antigen will be cloned (reproduced) independently and proportionally to their antigenic affinities, the higher the antigenic affinity, the higher the clones generated for each of the selected antigens. Based on the fact that the fittest antibody will produce more clones compared to weaker ones, the following equation for adaptive cloning process can be developed. The number of clones is generated according to the affinity measure or the fitness value using the following equation:

$$\text{No. of clones for each of the antigens} = \sum_{i=1}^{N_p} \text{round}\left(\frac{\beta * N}{i}\right)$$

Here, β = multiplying factor, N = total number of antibody and N_p = selected number of antibodies.

Mutation

Binary string is used to represent the attributes of the

Table 1. Computational results for n=20.

Problem		Best solution obtained by NEH heuristic		Best solution obtained by HC heuristic		Best solution obtained by PCSA	
n	S	ET_{NEH}	CPT TIMES (s)	ET_{HC}	CPT TIMES (s)	ET_{PCSA}	CPT TIMES (s)
	2	2236	5.23	2456	6.21	2135	4.45
	2	2489	5.23	2564	6.32	2348	4.65
20	2	1846	5.23	2156	6.11	1489	3.87
	2	2135	6.21	2368	5.78	2015	3
	2	2835	6.21	3489	5.64	2751	3
Average		2308.2	5.62	2606.6	6.01	2147.6	3.79
	5	4568	11.2	5236	12.58	4512	9.24
	5	5689	12.68	5489	14.25	3254	8
20	5	3568	10.48	4512	13.68	2899	9.27
	5	4896	13.58	6253	10.59	3877	10
	5	4125	14.25	5781	11.65	3018	10
Average		4569.2	12.44	5454.2	12.55	3512	9.30
	8	5028	21	5567	24	4895	18
	8	6212	22.03	7894	25.18	4789	17.54
20	8	1028	18.92	8025	23.21	8562	18.69
	8	5689	19.28	1235	19.36	4789	19.24
	8	4781	22	7814	20	3578	20
Average		4547.6	20.65	6107	22.35	5322.6	18.69

antibodies. After decoding, each antibody attribute will be in a form of pair of real valued vector (PT_j, η_j) $\forall j\{1, \ldots, n\}$where η_j is a strategy parameter. Each antibody will go through the mutation process according to the expression given by Equations (4) and (5).

$$\eta_j'^{(j)} = \eta_j'^{(j)} \exp(\tau'(N(0,1) + \tau N_i(0,1)) \quad (4)$$

$$PT_j'^{(j)} = PT^{(j)} + \eta_j'^{(j)} N_i(0,1) \quad (5)$$

Where, N (0, 1) is a normally distributed random number with zero mean and standard deviation equal to one. $N_i(0, 1)$ is a random number newly generated for every i and j. The factors $\tau = ((2(n_p)^{1/2})^{1/2})$ _1 and $\tau' = ((2n_p)^{1/2})$ _1 are commonly known as learning rates. An offspring PT'_i is calculated by Gaussian mutation.

Selection

When the mutation is over, the new set of antibodies thus obtained will satisfy the final storage volume constraints. Then the solution with lesser cost, that is, higher affinity values or fitness values are replaced by the initial population of antibodies. With the members of the next generation thus selected, the process repeated until the maximum number of generations reaches. It is assumed that the highest affinities are sorted in an ascending order. In selection, the offsprings produced by mutation process are sorted and the best value from the offspring and parents is calculated.

COMPUTATIONAL EXPERIMENTS AND RESULTS

We show the results of a series of computational experiments conducted to test the effectiveness of the proposed approach. Our algorithms are coded in C++ and all tests are conducted on a laptop with Pentium IV Core 2 Duo 2.53 GHz CPU. The computational tests are performed on randomly generated benchmark problems. The problem sizes are a combination of n= (20,50,100) jobs and S=(2,5,8) stages. For each combination of n and S, five instances are generated with job processing times p_{ij} uniformly distributed over [1,100] since the processing times of most library benchmark problem are generated in this range. Then, the number of machines m_i at stage i was chosen randomly over [1, 5]. The due date d_j of each job j is determined by the following equation:

$$d_j = \sum_{i=1}^{n} p_{ij}(1 + z) \quad 6)$$

Where c $\in$ [0-1]

For each problem, a set of 10 replicates is done and the best objective value of these replications is kept. Tables 1, 2 and 3 compares the sum of the total earliness and

Table 2. Computational results for n=50.

Problem		Best solution obtained by NEH heuristic		Best solution obtained by HC heuristic		Best solution obtained by PCSA	
n	S	ET_{NEH}	CPT TIMES (s)	ET_{HC}	CPT TIMES (s)	ET_{PCSA}	CPT TIMES (s)
	2	10548	41.25	11525	47.21	8796	38.25
	2	12354	43.56	12358	45	9654	35.71
50	2	18425	38.51	8025	44.69	11235	36.24
	2	12296	37.24	12735	47.23	1025	39.86
	2	9548	45	11191	49	9548	41
Average		12634.2	41.11	13166.8	46.63	8051.6	38.21
	5	21458	80.06	24236	89.76	17546	75.03
	5	23568	80	25895	90.49	18212	74
50	5	29564	79.28	32569	92.31	17239	73.25
	5	25326	82.86	26356	90	18957	76.54
	5	21478	82	23254	90	19021	78
Average		24278.8	80.84	26462	90.51	18195	75.36
	8	41235	75	45652	86	35652	70
	8	45236	74.23	49865	92.23	36259	69.28
50	8	51236	72.03	51228	91.95	35669	74
	8	48965	89	49856	93	34235	75.26
	8	43569	82	46985	93	39875	78.57
Average		46048.2	78.45	48717.2	91.23	36338	73.42

Table 3. Computational results for n=100.

Problem		Best solution obtained by NEH heuristic		Best solution obtained by HC heuristic		Best solution obtained by PCSA	
n	S	ET_{NEH}	CPT TIMES (s)	ET_{HC}	CPT TIMES (s)	ET_{PCSA}	CPT TIMES (s)
	2	90458	50.23	100236	51.84	87265	40,21
	2	91235	49.32	89567	50	88568	42
100	2	92356	48.25	112365	49.59	90218	43.58
	2	89452	49	98564	52	87562	44.89
	2	88742	50	87965	54,21	89572	47
Average		90448.6	49.36	97739.4	51.53	88637	43.57
	5	114562	70	120365	82	118235	45
	5	120325	71.26	124235	81,23	95628	48.26
100	5	124325	72.35	130568	83,26	98657	47.25
	5	118568	73	130589	84.25	102365	42.28
	5	128654	73.51	129658	85	96324	49.25
Average		121286.8	72.024	127083	83.148	102241,8	46.408
	8	170235	90	180235	102	153268	53
	8	152365	89	181235	96	145236	52.99
100	8	148956	87.25	164235	95.24	132589	54
	8	175623	86.32	152369	97,54	148523	56.08
	8	175423	84.26	154899	96	133259	51
Average		164520.4	87.366	166594.6	97.356	142575	53.414

tardiness values overall jobs for the best solution obtained by the NEH heuristic and the HC heuristic and the PCSA method for n=20,50 and 100 consecutively.

It is clearly shown in those table shows that PCSA outperform more than the NEH heuristic and the HC heuristic.

As we expected, the computation time (CPU times in seconds) is reasonable. We also observe the increasing of the CPU when the problem size increases. CPU times for PCSA are better than NEH and HC.

Conclusions

In this study, we propose a clonal selection algorithm (PCSA) to solve a HSF scheduling problem with regard to being NP-hard, the method of PCSA are coded in C++ and then the quality of the results with their time of calculation is compared with the results obtained from NEH and HC and the experiments prove that the PCSA can be used to solve HFS effectively and the efficiency of PCSA has been perfectly shown by the expansion of the said model for other state of production such as parallel machine series machine more researchers could done for future works.

Other Meta heuristic methods like memetic algorithm, GA algorithm forbidden search algorithm could be used for the presented models as well.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the support provided by the Islamic Azad University, Tonekabon Branch and thanks them for their sincere cooperation and support during this research plan, that has resulted in this study.

REFERENCES

Akkerman R, van Donk D, Gaalman G (2007). Influence of capacity- and time- constraint intermediate storage in two-stage food production systems. Int. J. Product. Res., 45(13):2955–3073.

Almeida MT, Centeno M (1998). A composite heuristic for the single machine early tardy job scheduling problem. Comput. Oper. Res., 25: 535–625.

Finke DA, Medeiros DJ, Traband MT (2007). Multiple machine JIT scheduling: a tabu search approach. Int. J. Product. Res., 45(21): 4899–4915.

Grabowski J, Pempera J (2000). Sequencing of jobs in some production system. Eur. J. Oper. Res., 125(3): 535–550.

Ho JC, Chang YL (1991). A new heuristic for the n-job, m-machine flowshop problem. Eur. J. Oper. Res., 52: 194–202.

Kis K, Pesch E (2005). A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. Eur. J. Oper. Res., 164: 592–608.

Kumar A, Prakash A, Shankar R, Tiwari MK (2006) Psycho-Clonal algorithm based approach to solve continuous flow shopscheduling problem. Expert Syst. Appl., 31(3): 504–514.

Low C (2006). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Comput. Oper. Res., 32: 2013–2025.

Nawaz M, Enscore E, Ham I (1983). A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. OMEGA, 11(1): 91–95.

M'Hallah R (2007). Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. Comput. Oper. Res., 34: 3126–3142.

Oguz C, Ercan M, Edwin Cheng T, Fung Y (2011). Heuristic algorithms for multi- processor task scheduling in a two-stage hybrid flow-shop. Eur. J. Oper. Res., 110: 390–403.

Ruiz R, Serifoglu F, Urlings T (2008). Modeling realistic hybrid flowshop scheduling problems. Comput. Oper. Res., 35: 1151–1175.

Ruiz R, Vazquez-Rodriguez J (2010). The hybrid flow shop scheduling problem. Eur. J. Oper. Res., 205: 1–18.

Sawik T (2002). An exact approach for batch scheduling in flexible flow lines with limited intermediate buffers. Math. Comput. Model., 36: 461–471.

Su L (2003). A hybrid two-stage flowshop with limited waiting time constraints. Comput. Ind. Eng., 47: 409–424.

Wang Z, Xing W, Bai F (2005). No-wait flexible flowshop scheduling with no-idle machines. Oper. Res. Lett., 33(6): 609–614.

Wardono B, Fathi Y (2004). A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities. Eur. J. Oper. Res., 155: 380–401.

Yang D, Chern M (1995). A two-machine flowshop scheduling problem with limited waiting time constraints. Comput. Ind. Eng., 28(1): 63–70.

Yang JH, Sun L, Lee HP, Qian Y, Liang YC (2008) Clonal selection based memetic algorithm for job shop scheduling problems. J. Bionic Eng., 5(2): 111–119.

Ying K, Lin S (2009). Scheduling multistage hybrid flowshops with multiprocessor tasks by an effective heuristic. Int. J. Product. Res., 13(1): 3525–3538.