Algorithms of Minimizing Makespan and Range of Lateness of Flow Shop Machines

Adel Hashem Nouri
Department of Computer Science, College of Education, University of Kufa, Najaf, Iraq, adilh.alhajjar@uokufa.edu.iq

Hussam Abid Ali Mohammed
Department of Mathematics, College of Education for Pure Sciences, University of Karbala, Karbala, Iraq, hussam.abidali@uokerbala.edu.iq

Kareema Abed Al-Kadim
Department of Mathematics, College of Education for Pure Sciences, University of Babylon, Hilla, Iraq, kareema.kadim@yahoo.com

Abstract

In this paper, the three-machine variant flow shop scheduling problem is addressed with the bi-objectives of reducing the makespan (Cmax) and range of lateness (RL). The goal is to concatenating minimize the two objective functions, i.e., to minimize the range of lateness provided that the makespan is optimum. To discover the best solution to the issue for job sequences with up to 2000 jobs, we introduce the Branch and Bound algorithm, and use the genetic and the memetic algorithms to find pareto optimal solutions, and then the results will be compared between them.

Keywords: *Flowshop machine, makespan, range of lateness, Branch and Bound algorithm, genetic algorithm, memetic algorithm.*

Introduction

A scheduling problem in a flowshop contains n tasks that must be completed on m machines in the same machine-by-machine sequence. Passing is allowed in the flowshop for permutations. Therefore, the sequence in which different jobs visit a set of machines is the same. The general flowshop allows passage. As a result, the order of the jobs on each machine may differ.

Many papers on scheduling have treated the multi-objective flowshop scheduling problem. Marett and Wright (1996) used a large and sophisticated multi-objective flow shop quality measurement problem of simulated annealing and tabu search. Sayin and Karabati (1999) reduced the total time it took to complete the task and its makespan concurrently to solve the scheduling problem in a two machine flowshop system.

There are typically three packed structure of multiobjective scheduling problems. Tradeoffs between objective functions are allowed in the first class, and the ultimate result is the result of the minimization of an aggregation function of the studied objective functions. In the second class, the objective functions were arranged according to their relative importance (lexicographical order), with the first objective being minimized in proportion to its best value, followed by the second, and so on. In the third

class, the objectives were concurrently decreased, and a set of non-dominated solutions was generated.

Important Notations

n : Number of jobs.
p_j: Processing time of jobs j
d_j: Due date of jobs j.
L_j: Lateness of job j, $L_j = C_j - d_j$.
C_j : Completion time of job j, where $C_j = \sum_{k=1}^{j} p_k$.
R_L: Range of lateness, $R_L = L_{max} - L_{min}$.

Some Important Definitions

Johnson's 3 machine algorithm [12]:Optimal makespan in a three-machine flow shop may be obtained by extending Johnson's two-machine method. Either,
min (p1j)≥ max(p2j) or min(p3j)≥ max(p2j)
Palmer's Heuristic (PR) [12]:: The heuristic is minimizing of makespan (Fm | Cmax) . These are the two stages of this heuristic:

Step 1: Determine the inclination for the n-job, m-machine static flow shop issue. As follows, Aj, for the jth job:; Aj= - $\sum_{i=1}^{m}\{m-(2i-1)\}$ p ij
Step 2: Apply a descending (decreasing) order to the jobs in the sequence based on the Aj values.
Earliest Due Date (EDD)[12]: For each open job, the one with the lowest date is completed first.

Problem Formulation

Suppose we have a set of n jobs $N = \{J_1, J_2, \dots, J_n\}$ Let's assume we have a list of n jobs, N=J1, J2,..., Jn, ready to be processed at t=0, and that these tasks may be run in any sequence on machines A1, A2, and A3 during periods of uninterrupted processing. In this study, we concentrate on the makespan function, which aims to maximize both (C_{max}) and range of lateness (R_L).

$$
\left.
\begin{aligned}
& Min \begin{cases} f_1(s) = C_{max} \\ f_2(s) = R_L \end{cases} \\
& s.t. \\
& C_{i1} = \sum_{k=1}^{i} p_{k1} && i = 1, \dots, n, \dots\dots (1) \\
& C_{12} = p_{11} + p_{12}, && \dots\dots\dots (2) \\
& C_{i2} = max\{C_{i1}, C_{i-1,2}\} + p_{i2}, && i = 2, \dots, n, \quad \dots (3) \\
& C_{13} = p_{11} + p_{12} + p_{13}, && \dots\dots\dots (4) \\
& C_{i3} = max\{C_{i2}, C_{i-1,3}\} + p_{i3}, && i = 2, \dots, n, \dots\dots\dots (5) \\
& L_i = C_{i3} - d_i, && i = 1, \dots, n. \quad \dots\dots (6)
\end{aligned}
\right\} (P)
$$

Solution Approaches:

Branch and Bound Algorithm (BAB) [5]:
The COP may be solved precisely using Branch and Bound (BAB). It operates on the principle of compiling a comprehensive set of viable options in an intelligent manner. To elaborate, this is because the desired solution to the discrete optimization problem P is a minimization. Now, break P into smaller issues defined by smaller subsets of the set S of possible solutions. P and its subproblems may be quickly and easily associated with the matching subset $S' \subseteq S$. Three procedures are needed for a BAB algorithm.
1. Branching: The forward branching rule is used, let S is replaced by smaller subsets $S_i (i = 1, \dots, r)$ such that $S = \bigcup_{i=1}^{r} S_i$. The term "branching" describes this phenomenon. Each S i is the seed from which another branch grows; hence, branching is a cyclical process.

A branching tree is used to depict the full branching process. Upper and initial lower limits are determined for subsets of S, denoted by S i (i=1,...,r), while S represents the root of the tree. Subproblems are instances of discrete optimization that are brought up during the forking procedure.

2. Lower bounding (LB): If you're using an LB scheme, the LB will be linked to every branch in the index. The goal is to get rid of all of the nodes where the LB is higher than the value of the best known viable solution. It is possible to determine an LB for this issue P by dividing the sequence into subsequences $s(= s' \cup s'')$ into two subsequences, s' be a schedule jobs and s'' be unscheduled jobs such that $s_1'' = s''(JR)$, $s_2'' = s''(PR)$ and $s_3'' = s''(EDD)$, then the decomposition LB is given by

$$LB_1 = \left(C_{max}(s_1 \cup s_1''), R_L(s_1 \cup s_3'')\right),$$
$$LB_2 = \left(C_{max}(s_1 \cup s_2''), R_L(s_1 \cup s_3'')\right),$$
$$LB = max(LB_1, LB_2)$$

3. Upper bounding (UB): We get to a UB of P's actual value. Such a UB may be derived from the objective value of any practically viable solution.

Now, if a particular subproblem's LB is larger than or equal to UB, then that subproblem cannot provide a superior solution to the main problem P. Therefore, we may stop branching from the matching node in the tree. The bound UB should be as tiny as feasible and the bound LB should be as big as possible to prevent branching at as many nodes of the branching tree as possible. This is why we use certain heuristics at the beginning of the BAB algorithm to locate a suitable viable solution with a low value of UB. The primary action in the BAB algorithm is to locate a suitable LB to prune the BAB bushes. There may be a single viable solution to the subproblem after multiple branches have been taken. Next, we take the objective value of this solution and use it to replace the LB in the subproblem UB by LB if LB < UB, i.e. set UB = LB.

Genetic Algorithm

Based on the ideas of evolution and natural selection, the genetic algorithm (GA) is a method of optimization and searching. In a GA, many individuals make up a population, and that population evolves according to principles of selection that aim to optimize "fitness" (i.e., maximizes the benefit function). The figure below [11] displays a simplified version of a genetic algorithm, one of the most important tools in the field of artificial evolution. Starting with a seed group (population of parents). Children may be born with crossover and mutation. Children are bred to become parents and so on until evolution is halted. This framework may be tailored to the specifics of a given problem-solving situation.

The proposed genetic algorithm consists of a chromosome, which represents a solution, and two primary components, the sequence and the idle periods introduced at the beginning of the algorithm.For example [5] [4,1,2,3] shows that jobs 4, 1, 2, and 3 will be processed first, with jobs 1 beginning at time 6. These genes were utilized as genetic operators: Two distinct crossover operators have been built in. Firstly, there's the infamous crossover between Order and (OX) [6]. In this method, two parents are picked at random, and a little portion of one of their chromosomes is replicated into the embryo. During the second stage, the other parent's chromosomes are inserted into the resulting child in the order of their appearance. Mohammed et al. [8] proposed a second crossover they called homogeneous mixture crossover (HMX), which involves mixing the two parental chromosomes uniformly by creating a set of genes called M. They also presented a method for the mixture, which

involves first taking the odd position from the first parent and the even position from the second. Then, we can split the genes apart without repeating any of them; this is done by reading the set M from left to right, inserting gene j into the first child if it doesn't already exist there, and inserting it into the second if it does. An additional pair of chromosomes is produced in this manner.

• Mutation: Following this decision, in our implementation we use the classic mutation technique denoted by the Adjacent Pairwise Interchange (API) area. Description of the Roots of the GA using a chromosomal segment from each parent,

• Initialization: The first stage of GA involves randomly generating a large number of individual solutions. Several hundred solutions are often included in the population's first generation, however this number varies depending on the specifics of the situation at hand. Since the traditional method of population generation relies on a random number generator, it makes it possible for any and all solutions to emerge (the search space). It's possible to "seed" solutions in locations where they're more likely to be discovered successfully.

• Selection: Every new generation is made up of a chosen subset of the previous one. Fitter solutions (as assessed by a fitness function) are often more likely to be picked during the fitness-based solution selection process. Some techniques of selection apply a fitness rating to each solution and then choose the ones with the highest ratings. Because it may take a long time to evaluate the whole population, some techniques just evaluate a representative sample.

• Reproduction: Those solutions that do well in the genetic selection process will then be used to produce a new generation of solutions. A pair of "parent" solutions is chosen from the pool of preselected solutions to be used in the generation of a new solution. In general, when two solutions are crossed and mutated to create a "child," the offspring takes on many of the traits of both of its parents. Every new generation begins with a fresh set of parents, and this continues until a set of solutions of the right size has been produced. Some studies imply that more than two "parents" are better to reproduce a high quality kid [9], despite the fact that reproduction techniques based on the usage of two parents are more akin to the nature of biology. By acting in this way, the following generation produces a set of kids that are distinct from their parents. Since only the most fit creatures from the first generation are chosen to reproduce (along with a small percentage of less fit solutions, as indicated above), this approach often results in an increase in the population's average fitness. While crossover and mutation get all the attention, genetic algorithms may also make use of additional operators including migration, colonization, and even regrouping.

• Termination: This process of passing on characteristics from one generation to the next is continued until a certain threshold is met, usually measured in terms of the number of generations.

Simple generational genetic algorithm procedure:

• Select the first set of individuals.

• To determine how healthy each person is, you must examine the population as a whole.

• Continue with this generation forever (time limit, sufficient fitness achieved, etc.)

• Choose the healthiest and most fertile people.

• The process of producing offspring by use of crossover and mutation procedures.

• Assess the health and fitness of newcomers..

• The weakest members of the population should be swapped out for fresh ones.

Memetic Algorithm

Memetic algorithms [10] might be seen as the union of a population-based global strategy with a personalized local search. They are a kind of genetic algorithms known for their use of a hill-climbing local environment. Memetic algorithms are a kind of population-based AI similar to genetic ones. Some tests have revealed that their speed surpasses that of typical genetic algorithms by many orders of magnitude. A ternary tree-based population structure was selected for use in this memetic algorithm. It separates the people into groups and limits the opportunities for mixing, in contrast to a population that is not organized in any way.

Population structure

Each cluster in the structure has a leader and three follower solutions. The group choose its leader by identifying its most capable member. There must be at least 13 persons for a ternary tree with three levels, at least 40 for a ternary tree with four levels, and so on.

• Representation: Our representation for the permutation flow shop scheduling issue is rather natural; a solution is shown as a chromosome, with alleles taking on various integer values in the [1, n] range, where n is the number of tasks.

• Crossover: The crossovers in GA are the same as those mentioned up above..

• Mutation: The similar mutation in GA was already mentioned up above.

• Fitness Function: The fitness function was arbitrarily selected, since the objective in this challenge is to minimize the weighted mean completion time and weighted mean tardiness.

• Offspring Insertion in Population:

After a leader and follower are chosen, the reproductive processes of recombination, mutation, and local search begin, ultimately leading to the birth of a new generation. If a progeny's fitness levels are higher than those of the leader, the progeny takes over. However, if the recombination was successful, it will replace the original supporter. A new member of the population is not introduced if there is already an individual with that identifier in the population. For diversity's sake, we made it a policy to not permit identical persons. The populace is reorganized once all new members have been added. A group's fitness leader must have a lower score than the group's fitness leader immediately above them. If this strategy is followed, the root subgroup's leader will be the most fit of all the subgroup heads, while those at higher levels will have the least fit leaders. Each subgroup's leader is compared to the leader of the next higher subgroup to determine the necessary modification. In the event that the leader of the level below proves to be superior, the two leaders will switch positions.

Computational Experience

Test Problems

In this part, we conduct a series of experiments to demonstrate the efficacy of the aforementioned algorithm. The goal of these simulations is to evaluate the merits of the memetic algorithm technique vs the genetic algorithm approach to solving the Permutation Flowshop Scheduling Problem. The experiments were conducted on Pentium IV at

2.2GHz, 2GB computer using "Matlab" language.

In order to evaluate the algorithms' capabilities, a collection of test problems was developed. The number of machines and the number of open positions are two primary indicators of the scope of a problem. The effectiveness of algorithms that discover near-optimal solutions is likely to be impacted by the degree to which processing times for individual tasks are correlated. Using three machines and a total of 10, 20, 30, 40, 50, 75, 100, 150, 200, 500, 1000, and 2000 tasks, a representative sample of test issues was created. The processing times p i1, p i2, and p i3 used in the test problems were drawn at random from a uniform distribution on the integers given by the range [1,10], and the deadlines were also produced using a uniform distribution $\left[\left(1-TF-\frac{RDD}{2}\right)SP, \left(1-TF+\frac{RDD}{2}\right)SP\right]$ such that $SP=\sum_{I=1}^{n} k_i$ where $k_i=\frac{(p_{i1}+p_{i2}+p_{i3})}{3}$, TF = 0.2, 0.4, RDD = 0.2, 0.4, 0.6, 0.8, 1, and the due date generation follow that given in [7]. For each value of n jobs we have average 10 problems.

Comparative Results

Results from our computer experiments demonstrating the efficacy of our BAB and local search algorithms are shown here. Our next step is to evaluate the data we've collected with...

[Table 1 Compare among BAB,GA and MA]

N	BAB		GA		MA	
	Eff	Range of Time	Eff	Range of Time	Eff	Range of Time
5	19	0.00798	19	0.096545	19	0.110606
6	20	0.00559	19	0.096545	20	0.066379
7	26	0.00919	26	0.062949	26	0.066121
8	28	0.011839	25	0.065905	28	0.069294
9	26	0.014193	23	0.070522	23	0.071937
10	28	0.018243	22	0.076651	28	0.076276
15	30	0.044925	20	0.095055	28	0.095471
20	28	0.079979	22	0.112917	23	0.116034
30	31	0.217481	10	0.15431	10	0.155799
40	47	0.629891	10	0.192842	10	0.192815
50	45	1.043772	8	0.228183	8	0.234993
75	49	3.613651	9	0.331251	9	0.334316
100	48	7.600067	8	0.468497	8	0.435716
200	46	58.01937	11	1.423861	11	0.832528
500				3.570555		2.069793
1000				4.695768		7.777485
2000				10.64744		10.85984

N	GA		MA	
	Eff	Range of Time	Eff	Range of Time
5	19	0.096545	19	0.110606
6	19	0.096545	20	0.066379
7	24	0.062949	26	0.066121
8	25	0.065905	28	0.069294
9	23	0.070522	23	0.071937
10	22	0.076651	28	0.076276
15	20	0.095055	28	0.095471
20	22	0.112917	26	0.116034
30	10	0.15431	23	0.155799
40	10	0.192842	33	0.192815
50	8	0.228183	22	0.234993
75	9	0.331251	22	0.334316
100	8	0.468497	22	0.435716
200	11	1.423861	14	0.832528
500		3.570555		2.069793
1000		4.695768		7.777485
2000		10.64744		10.85984

Table 2 below compares the time and value spent on each of two popular local search heuristics: the genetic algorithm (GA) and the memetic algorithm (MA).

[Table 2]

Number of jobs	I	BAB	time	N.S	GA	time	N.S	MA	time	N.S
5	1	(143,101),(146,93)	0.0496	2	(143,101),(146,93)	0.0965	2	(143,101),(146,93)	0.11060	2
	2	(148,89),(158,86)	0.0039	2	(148,89),(158,86)	0.0965	2	(148,89),(158,86)	0.11060	2
	3	(139,62)	0.0023	1	(139,62)	0.0965	1	(139,62)	0.11060	1
	4	(143,110),(148,88), (149,75),(150,54)	0.0062	4	(143,110),(148,88) ,(149,75),(150,54)	0.09654	4	(143,110),(148,88) ,(149,75),(150,54)	0.11060	4
	5	(133,91),(139,87), (143,55)	0.0047	3	(133,91),(139,87), (143,55)	0.09654	3	(133,91),(139,87), (143,55)	0.11060	3
	6	(131,80)	0.0020	1	(131,80)	0.09654	1	(131,80)	0.11060	1
	7	(136,62)	0.0018	1	(136,62)	0.09654	1	(136,62)	0.11060	1
	8	(154,77)	0.0022	1	(154,77)	0.09654	1	(154,77)	0.11060	1
	9	(147,47)	0.0024	1	(147,47)	0.09654	1	(147,47)	0.11060	1
	10	(146,96),(149,61), (155,51)	0.0047	3	(146,96),(149,61), (155,51)	0.09654	3	(146,96),(149,61) ,(155,51)	0.11060	3
6	1	(168,122;173,121;174 ,116)	0.0095	3	(168,122;173,121; 174,116)	0.05791	2	(168,122),(173,12 1),(174,116)	0.07046	3
	2	(163,100;164,94)	0.0052	2	(163,100;164,94)	0.05791	2	(163,100;164,94)	0.07046	2
	3	(176,131;180,94)	0.0052	2	(176,131;180,94)	0.05791	2	(176,131;180,94)	0.07046	2
	4	(163,82;168,64)	0.0047	2	(163,82;168,64)	0.05791	2	(163,82;168,64)	0.07046	2
	5	(167,81;178,74)	0.0047	2	(167,81;178,74)	0.05791	2	(167,81;178,74)	0.07046	2
	6	(158,102)	0.0039	1	(158,102)	0.05791	1	(158,102)	0.06229	1
	7	(165,92)	0.0037	1	(165,92)	0.05791	1	(165,92)	0.06229	1
	8	(165,116;168,110;171 ,89)	0.0072	3	(165,116;168,110; 171,89)	0.05791	3	(165,116;168,110; 171,89)	0.06229	3
	9	(170,80;172,56)	0.0065	2	(170,80;172,56)	0.05791	2	(170,80;172,56)	0.06229	2
	10	(162,83;164,58)	0.0053	2	(162,83;164,58)	0.05791	2	(162,83;164,58)	0.06229	2
7	1	(185,139),(189,134) ,(191,133),(196,131)	0.0165	4	(185,139),(189,134),(19 1,133),(196,131)	0.06294	4	(185,139),(189,134), (191,133),(196,131)	0.0662	4
	2	(189,128),(201,122), (203,117)	0.0115	3	(189,128),(201,122),(20 3,117)	0.06294	3	(189,128),(201,122), (203,117)	0.06610	3
	3	(188,146),(195,132), (198,91)	0.0126	3	(188,146),(195,132),(19 8,91)	0.06294	3	(188,146),(195,132),(19 8,91)	0.06610	3
	4	(185,111),(187,69)	0.0065	2	(185,111),(187,69)	0.06294	2	(185,111),(187,69)	0.06610	2
	5	(199,74)	0.0052	1	(199,74)	0.06294	1	(199,74)	0.06610	1
	6	(184,132),(190,126)	0.0062	2	(184,132),(190,126)	0.06294	2	(184,132),(190,126)	0.06610	2
	7	(194,135),(195,126), (204,112)	0.0094	3	(194,145)(195,136) (204,112)	0.06294	1	(194,135),(195,126), (204,112)	0.06610	3
	8	(188,105),(196,97)	0.0062	2	(188,105),(196,97)	0.06294	2	(188,105),(196,97)	0.06610	2
	9	(195,149),(202,130;20 4,111)	0.0090	3	(195,149),(202,130),(20 4,111)	0.06294	3	(195,149),(202,130),(20 4,111)	0.06610	3
	10	(201,82),(207,69), (215,52)	0.0088	3	(201,82),(207,69);(215, 52)	0.06294	3	(201,82),(207,69), (215,52)	0.06610	3
8	1	(218,158),(222,157)	0.0095	2	(218,158),(222,157)	0.06590	2	(218,158),(222,157)	0.06868	2
	2	(212,145)	0.0049	1	(212,145)	0.06590	1	(212,145)	0.06868	1
	3	(220,175),(226,128) ,(228,109)	0.01339	3	(220,178)(226,131) (228,109)	0.06590	1	(220,175),(226,128) ,(228,109)	0.06868	3
	4	(208,144),(209,102), (211,99),(212,88)	0.0155	4	(208,144),(209,1020), (211,99),(212,88)	0.06590	4	(208,144),(209,102), (211,99),(212,88)	0.06868	4
	5	(228,168),(229,102) ,(241,90)	0.0123	3	(228,168)(229,117), (231,112)(241,90)	0.06590	2	(228,168),(229,102), (241,90)	0.06868	3
	6	(217,171),(218,170) ,(219,158)	0.0120	3	(217,171),(218,170), (219,158)	0.06590	3	(217,171),(218,170) ,(219,158)	0.06868	3
	7	(224,182),(227,142) ,(234,130)	0.0129	3	(224,182),(227,142), (234,130)	0.06590	3	(224,182),(227,142), (234,130)	0.06868	3
	8	(223,187),(225,158), (226,141),(228,121)	0.0159	4	(223,187),(225,158), (226,141),(228,121)	0.06590	4	(223,187),(225,158), (226,141),(228,121)	0.07070	4
	9	(220,161),(221,150), (222,115)	0.0130	3	(220,161),(221,150), (222,115)	0.06590	3	(220,161), (221,150),(222,115)	0.07070	3
	10	(233,122),(240,118)	0.0090	2	(233,122),(240,118)	0.06590	2	(233,122),(240,118)	0.07070	2
9	1	[242,180),(247,175]	0.0108115	2	(242,180),(247,175)	0.07052	2	(242,180),(247,175)	0.07253	2
	2	[249,184),(250,157]	0.011306	2	(249,184),(250,157)	0.07052	2	(249,184),(250,157)	0.07253	2
	3	[251,177),(253,144), (262,142]	0.0158042	3	(251,177),(253,144) ;(262,142)	0.07052	3	(251,177),(253,1440, (262,142)	0.07253	3
	4	[253,148),(256,126]	0.0112437	2	(253,148),(256,126)	0.07052	2	(253,148),(256,126)	0.07167	2
	5	[252,140),(253,127]	0.010982	2	(252,140),(253,127)	0.07052	2	(252,140),(253,127)	0.07167	2
	6	[243,186),(247,182), (;252,179]	0.0160614	3	(243,186),(247,182) ,(252,179)	0.07052	3	(243,187),(247,183), (252,180)	0.07167	0
	7	[248,194),(250,181), (251,172]	0.0167563	3	(248,194),(250,181) ,(251,172)	0.07052	3	(248,194),(250,181) (251,172)	0.07167	3

	8	[248,218),(250,192),(252,175),(254,145),(261,126]	0.026801	5	(248,218),(250,205),(252,188),(254,158),(261,126)	0.07052	2	(248,218),(250,192),(252,175),(254,145),(261,126)	0.07167	5
	9	[245,150),(251,108]	0.0114249	2	(245,150),(251,108)	0.07052	2	(245,150),(251,108)	0.07167	2
	10	[261,159),(264,102]	0.0107432	2	(261,159;264,102)	0.07052	2	(261,159),(264,102)	0.07167	2
10	1	(279,215),(283,212),(287,207)	0.0213817	3	(279,219),(283,216);(287,211)	0.07665	0	(279,215),(283,212),(287,207)	0.07627	3
	2	[265,208),(269,177),(274,173]	0.0198716	3	(265,208),(269,177),(274,173)	0.07665	3	(265,208),(269,177),(274,173)	0.07627	3
	3	[291,162]	0.0077923	1	(291,162)	0.07665	1	(291,162)	0.07627	1
	4	[255,224),(258,137]	0.0138764	2	(255,224),(258,137)	0.07665	2	(255,224),(258,137)	0.07627	2
	5	[255,195),(258,102),(263,82]	0.0191021	3	(255,196),(258,102),(263,82)	0.07665	2	(255,195),(258,102),(263,82)	0.07627	3
	6	[262,208),(267,197),(272,191]	0.0190555	3	(262,208),(267,197),(272,191)	0.07665	3	(262,208),(267,197),(272,191)	0.07627	3
	7	[274,218),(275,182]	0.0134121	2	(274,218),(275,182)	0.07665	2	(274,218),(275,182)	0.07627	2
	8	[271,239),(275,199),(278,171),(283,146]	0.0244211	4	(271,239),(275,202),(278,174),(283,146)	0.07665	2	(271,239),(275,199),(278,171),(283,146)	0.07627	4
	9	[267,193),(268,174),(270,124]	0.0189098	3	(267,193),(268,174),(270,124)	0.07665	3	(267,193),(268,174),(270,124)	0.07627	3
	10	[272,183),(273,158),(277,132),(281,107]	0.0246089	4	(272,183),(273,158),(277,132),(281,107)	0.07665	4	(272,183),(273,158),(277,132),(281,107)	0.07627	4
15	1	[413,326]	0.0167295	1	(413,326)	0.09505	1	(413,326)	0.09585	1
	2	[399,331),(403,298),(406,270]	0.0447385	3	(399,339),(403,306),(406,270)	0.09505	1	(399,331),(403,298),(406,270)	0.09585	3
	3	[410,347),(417,272),(418,255),(419,253]	0.0586964	4	(410,347),(417,272),(418,259),(419,253)	0.09505	4	(410,347),(417,272),(418,255),(419,253)	0.09585	4
	4	[389,364),(390,202),(400,172]	0.0446966	3	(389,364),(390,221),(400,175)	0.09505	1	(389,364),(390,202),(400,175)	0.09585	2
	5	[401,224),(407,172),(410,169]	0.045394	3	(401,224),(407,172),(410,169)	0.09505	3	(401,224),(407,172),(410,169)	0.09585	3
	6	[402,353),(403,317]	0.0305985	2	(402,353),(403,317)	0.09505	2	(402,353),(403,317)	0.09585	2
	7	[409,354),(410,299),(413,287),(420,284]	0.0591481	4	(409,354),(410,308),(413,297),(420,284)	0.09505	2	(409,354),(410,299),(413,287),(420,284)	0.09585	4
	8	[415,323),(416,307),(417,244),(421,241),(422,231]	0.0730173	5	(415,343),(416,327),(417,264),(421,241),(422,231)	0.09505	2	(415,343),(416,307),(417,244),(421,241),(422,231)	0.09458	4
	9	[411,260),(415,213),(417,201]	0.0450706	3	(411,260),(415,213),(417,201)	0.09505	3	(411,260),(415,213),(417,201)	0.09458	3
	10	[381,253),(384,144]	0.0311645	2	(381,258),(384,144)	0.09505	1	(381,253),(384,144)	0.09458	2

Conclusion

In this paper, we consider that the three-system problem is reduced in lexical order The two objective functions, Makepan(Cmax) and range of lateness (RL), the problem denotes F3 || (Cmax, RL). We suggest several Algorithms for solving the problem, genetic and memetic algorithm. We also tried to improve the performance of the BAB algorithm by reducing the number of visits Contracts and calculation times for efficient solutions n = 2000. Future research can be done through an app suggested algorithms in practical production scheduling problems.

Reference

[1]J. M. Framiñán, R. Leisten, and R. R. Garc, "Manufacturing scheduling systems,

an integrated view on models," Methods and Tools, (2014) pp. 51–63.

[2] S. M. Johnson, "Optimal two-and three-stage production schedules with setup times

included," Nav. Res. Logist. Q., vol. 1, no. 1, (1954) pp. 61–68.

[3] Pinedo M. Scheduling: theory algorithms and systems. Englewood Cliffs, Prentice-Hall, New Jersey (1995).

[4] Solimanpur M., Vrat P. and Shankar R., A neuro-tabu search heuristic for flowshop scheduling problem. Computers & Operations Research 31: (2004): 2151–2164.

[5] Pinedo, L.M., Scheduling: Theory, Algorithms, and Systems, Fourth Edition Springer Science+Business Media, LLC, (2012).

[6] Mohammed H. A., Cheachan H. A. and Khtan Q. A.,: Single machine scheduling to minimizing sum penalty number of late jobs subject to minimize the sum weight of completion time. Journal of Kerbala University. 7(1): (2009)163–173.

[7] Yousefi M. and Yusuff R. M., Minimizing earliness and tardiness penalties in a single machine scheduling against common due date using genetic algorithm. Research Journal of Applied Sciences, Engineering and Technology 4(9): (2012) 1205-1210.

[8] Mohammed H. A., Hassan A. S., Saloomi M. H. and Khtan Q. A., Memetic Algorithm and Genetic Algorithm for the Single Machine Scheduling Problem with Linear Earliness and Quadratic Tardiness Costs. Journal of Kerbala University. 7(1): (2012): 163–173.

[9] Ting, Chuan-Kang): On Mean Convergence Time of Multi-parent Genetic Algorithms without Selection. Advances in Artificial Life, pp: 403-412. ISBN 978-3-540-28848(2005)

[10] Murata T., Ishibuchi H. and Tanaka H.,: Multi-objective genetic algorithm and its applications to flowshop scheduling. Computers and Industrial Engineering 30(1996) 957–968.

[11] Cheachan H. A., Mohammed H. A. and Khtan Q. A.: Scheduling flowshop machines to minimize the multi-objective functions. Iraqi Journal for Administrative Sciences (2010)637–657.

[12]Alharkan I. M., "Algorithms for Sequencing and Scheduling", Industrial Engineering Department, College of Engineering, King Saud University, Riyadh, Saudi Arabia, (2007).