International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org/journals.php
Vol. 7 Issue 5, May-2017, pp. 286~299
ISSN (O): 2249-3905, ISSN(P): 2349-6525 | Impact Factor: 7.196 | Thomson Reuters Researcher ID: L-5236-2015

Minimizing the Total Completion Times and Range of Lateness on a Single Machine by using Genetic Algorithms

Mohanad R. Aljanabi

Faculty of computer science and mathematics,Kufa university,Iraqi .

Abstract:

In this page, the main work is minimizing the function of two gauge of the scheduling n jobs on a single machine. Each job i requires a given processing time p_i and has a given due date d_i .The problem is to find a schedule that minimizes a function of total flowtimes and range of lateness. We proposed Genetic algorithm to find the set of efficient solutions to this problem . Computational results demonstrate the efficiency of these algorithm.

Keywords: Multicriteria optimization, total flowtimes, range of Lateness, efficient solutions, Genetic algorithm.

1.Introduction

The theory of scheduling has been evolv to find asulution for problems occuring in for instance facitities the production . The main problem of the scheduling is occure as finding for each of the tasks . And also can called jobs . The execution period that taken on one of the machines of the all side comstraints . This and the resulting of the solves must be happence in the same way wich it called schedulling . The minimizing the function of the objective that offer , is best possible , in 1954 the date of appearing the first schudelling paper . By differentiating between side -constraints , machine environment and objective function . Alotof various main scheduling problems have been formulated . [8].

Smith (1956)[12] solved the problem $1//(\sum C_i , T_{max})$ subject to minimum value of T_{max} by using Smith backward algorithm. Van wassenhove and Gelder (1980) extended this problem to $1//(\sum C_i , T_{max})$ problem, where T_{max} (EDD) ≠ 0. They suggested a pseudo polynomial algorithm for the problem. They used the modified due date concept and modified Smith backward algorithm to find the set of all efficient solutions. Tapan Sen et al.[14] presented a BAB algorithm to solve the $1// (\sum C_i + R_L)$ problem. They solved the problem with n≤15, where a liner combination of the two objectives is considered. Abdul-Razaq T. and Alshekhly S.(2016)[2]solve the $1//(\sum C_i , R_L)$ problem to find the set of efficient solutions by using some algorithms. The Genitic Aligarithm (GA) have been applied heuristic varied problem of the objective . [9] a Multi -Objective Genitic Aligarithm (MOGA) proposed by Murata et al , which it used in flowshop scheduling . The chosen procedure (in MOGA chosen single with different weights) , wich are generatic in each generation randomly . The paper is organized as follows. In section 2 we present the simultaneous minimization of the performance measure total

flowtimes and range of lateness, its mathematical form and characterizing efficient solutions for this problem. At the

beginning, an introduction to evolutionary algorithms (**Genetic Algorithms (GAS))** is presented in Section 3.In section 4 we present The Basic GA's cycle. In section 5 we present the describe structure of (GA) for the problem $1//(\sum C_i , R_L)$to find the approximate set of efficient solutions SA. Computational experience with genetic algorithms are given in section 6. In section 7 conclusions is given.

2.The mathematical form of the $1//(\sum C_i , R_L)$ problem[2]

The multicriteria scheduling problem is studied , concerns the simultaneous and hierarchical minimization of the performances measure total flowtimes ($\sum C_i$) and range of lateness (R_L). The mathematical forms and algorithms are given below.

First, for the $1//(\sum C_i , R_L)$ problem (P), we will try to find the set of efficient solutions, which can be written for a given schedule s = (1,…,n) as:

$$\text{Min} \begin{cases} \sum C_i(s) \\ R_L(s) \end{cases}$$

$$s \in S$$

s.t. …(P)

$$C_i \geq p_i \quad , \qquad i = 1,\ldots,n$$

$$C_i = C_{i-1} + p_i \, , \qquad i = 2,\ldots,n$$

$$L_i = C_i - d_i \quad , \qquad i = 1,\ldots,n$$

$$R_L(s) = L_{max}(s) - L_{min}(s)$$

This problem (P) is hard to solve and find a collection of all ective (Pareto optimal) solutions (SE).

In this problem the total flowtime (total completion times) and the range of lateness are used as the two criteria. The first object is to minimize flowtime (a measure for average in processing inventory). The second object deals with service for the evaluation of equal treatment to customers.

3.Genetic Algorithms (GAS):-

Genetic algorithms are global search and optimization techniques modeled from natural genetics. J.H. Holland in 1975 first described a GA, which is

commonly called the classical genetic algorithm(CGA).

The random - technique of the GA search wich several alogarithm carify natural phenomena (biological evolution) . The main thought of GA is that the strong moving to cope and survive and the weak moving to die . One of the strengths of GAs is that they are using previous information to direct their search with important performance assumption . The Grefenstette supplied the GA formal detailling as the follows: [3].

Agenitic Alogarithm is repeated procedure safeguard a population of structure that are elect solution to certain scope. The stream of the population structures are estimate of their effectiveness area through interim increasing (called a generation) . And on the basic of theee ratings. A new population of yhe offer solution is formative using particular operators of the genitic as mutation and crossover .

3.1.Initialization[3]:-

The size of the population is the first element and how to generate the initial population. Chromosome initial population is generating by using of some heuristic can fitness the considered problem or can be randomly generated . In the GAs the limitation of the population size is animporatnt component . Rising the risk prematurely telative to a local optimal through choose a very small size of the population . Large size of the population rise the chance to relative to a global optimal . But it will extra time to relative the size of the population was preserved at a fixed . In the most of the GA .

3.2. Fitness[4]:-

The fitness of individuals in the current population is evaluated chromosomes with higher fitness values tend to have more of their copies at the next generation [4]. The fitness function is often based on the objective functions.

3.3.Selection

Select a set of solutions P^* from $P \cup P_1$, and set $P = P^*$. GA choose good candidate solution from P for the next generation. Among different types of chosen strategies have been proposed [16], here we are using two simple strategies tn the following .

1.With a solution σ_{worst} satisfying cost (σ_{worst}) ≥ cost (σ) for all $\sigma \in$ P, let $P^* = P \cup \{ \sigma_{ck} \} - \{ \sigma_{worst} \}$ (k = 1, 2).

1. Suppose cost (σ_{p1}) ≤ cost (σ_{p2}) without loss of generality, and let $P^* = P \cup \{ \sigma_{ck} \} - \{ \sigma_{p2} \}$ (k = 1, 2) with probability.

$$Pr = \begin{cases} 1, & \text{if} \quad \text{cost}(\sigma_{ck}) \le \text{cost}(\sigma_{p2}), \quad k = 1,2 \\ \dfrac{\Delta p}{\Delta p + \Delta c} \end{cases}$$

Where Δp = cost (σ_{p2}) – cost (σ_{p1}), Δc = cost (σ_{ck}) – cost (σ_{p1}) (k = 1, 2); P^* = P with probability 1 – *Pr*.

3.4.Crossovers[3]:

The crossover operator which merge the feature of two matching chromosome and hold these features to next generation by creat offspring . When chosen two chrmosomes and undefined passing position (single - position cross method) then the SGA represent the crossover . Then the rendevous parental alternative to creat two nwe children . Alot of cross over methods have been evolved and used to binary .

- **Linear Order crossover (LOX):-**

Within LOX operator, two crossover points are selected, the elements in the cross section of the first parent are removed from the second parent leaving some holes. These holes slide from the extremities towards the center until they reach the cross section. The cross section is then substituted with that of parent 1. The other child is obtained similarly [6].

Example:-

	Parents		Holes
Parent 1	7 9 8 : 2 5 1 : 6 3 4	⇒	7 9 • : 2 5 1 : 6 • •
Parent 2	9 5 6 : 4 8 3 : 2 7 1		9 • 6 : 4 8 3 : • 7 •

	Sliding		Exchanging
	7 9 2 : • • • : 5 1 6	⇒	7 9 2 : 4 8 3 : 5 1 6
	9 6 4 : • • • : 8 3 7		9 6 4 : 2 5 1 : 8 3 7

As already mentioned in a previous paragraph, whenever a permutation representation is used, an appropriate crossover operator has to be devised. *Partially matched crossover* (PMX) is such an operator, introduced by Goldberg and Lingle [7] for the traveling salesman problem. Two crossover points are generated at random and the segments in between define a matching section. This matching is used to effect a cross through position-by-position exchange operations. For example, with crossover points after the 3*rd* and 6*th* element:

Parents:		Exchanging:		Restoring:
7 9 8 | 2 5 1 | 6 3 4	⇒	7 9 8 | 4 8 3 | 6 3 4	⇒	7 9 5 | 4 8 3 | 6 1 2
9 5 6 | 4 8 3 | 2 7 1		9 5 6 | 2 5 1 | 2 7 1		9 8 6 | 2 5 1 | 4 7 3

In this example, the mapping is $2 \leftrightarrow 4$, $5 \leftrightarrow 8$ and $1 \leftrightarrow 3$. Between the two crossover points, the sections are exchanged. To restore feasibility in the first child (σ_{c1}), the elements 8, 3 and 4 outside the section are replaced according to the matching. In the second child (σ_{c2}), the elements 5, 2 and 1 are replaced. A lot of papers refer to PMX, for example Chen *et al.* [5] and Nordström and Tufekci [10].

Now, in this section we introduce new two types of crossovers; first one similar to the LOX we call it *jumping order crossover* JOX, since in LOX in the stage (3) sliding the genes, here we jumping the genes that is gives new child different from the LOX, since;

Parents:		Holes:		Jumping:		Exchanging:
7 9 8 | 2 5 1 | 6 3 4	⇒	7 9 ◦ | 2 5 1 | 6 ◦ ◦	⇒	7 9 2 | ◦ ◦ ◦ | 6 5 1	⇒	7 9 2 | 4 8 3 | 6 5 1
9 5 6 | 4 8 3 | 2 7 1		9 ◦ 6 | 4 8 3 | ◦ 7 ◦		9 4 6 | ◦ ◦ ◦ | 8 7 3		9 4 6 | 2 5 1 | 8 7 3

The new crossover is similar to LOX so that in order to tend mainly to respect relative positions between the elements and also, as far as possible, the absolute positions in the string. In the first parent (σ_{p1}) the gene 8 replaced by 2, 3 replaced by 5 and 4 replaced by 1 and the similar way for the second parent (σ_{p2}), hence we obtain a new two child.

The second crossover calls *homogeneous mixture crossover* HMX given by the mixture the two parents uniformly by make a set from genes M, the odd position from the first parent and the even position from the second parent. Then separate genes without repetition gene, since we read the set M from the left, if the gene j does not existing in the child σ_{c*} put it, otherwise we put gene j in the second child $\sigma_{c\#}$ until final M. This way also gives a new two child.

Parents:		Mixture:		Exchanging:
7 9 8 2 5 1 6 3 4	⇒	7 9 9 5 8 6 2 4 5 8 1 3 6 2 3 7 4 1	⇒	σ_{c*}: 7 9 5 8 6 2 4 1 3
9 5 6 4 8 3 2 7 1		* * # * * * * * # # * * # # # # # #		$\sigma_{c\#}$: 9 5 8 6 2 3 7 4 1

3.5.Mutation:-

In his scheduling study, Syswerda [13] also introduces a number of mutation operators. In such an operator, two elements are selected at random. Order-mutation interchanges these two elements. Position-mutation places the second element before the first. Order-mutation performs better than position-mutation.

Tate and Smith [15] They select two locations at random in a string and reverse the order of all elements within the substring bounded by the two selected elements.

Other researches use the same techniques, but refer to them by more classical names, which are also used in neighbourhood search methods. Chen *et al.* [5] and Nordström and Tufekci [10] "swap" the two elements. Reeves [11] makes a few experiments with both operators. "Shift" seems to be better than "swap" and therefore "shift" is adopted in his final version .

3.6.Termination[3]

Stopping creation is the final component in the GA methods . Many gauge that was offored . One of them is that if achived the maximum number of the generation or if the population has converge then the GA will asop . The population convergence was explained by many gauge of the reasearchers . One of these is GA converges after a chromosome to comfirmed high -fitness rate is located . And also all chromosomes achived appointed level of homogeneity (that is , all of them have mostly the similar fitness amount) .

4.The Basic GA's cycle

The basic GA cycle based on the three processes (selection, mating and mutation) as shown in figure (1).

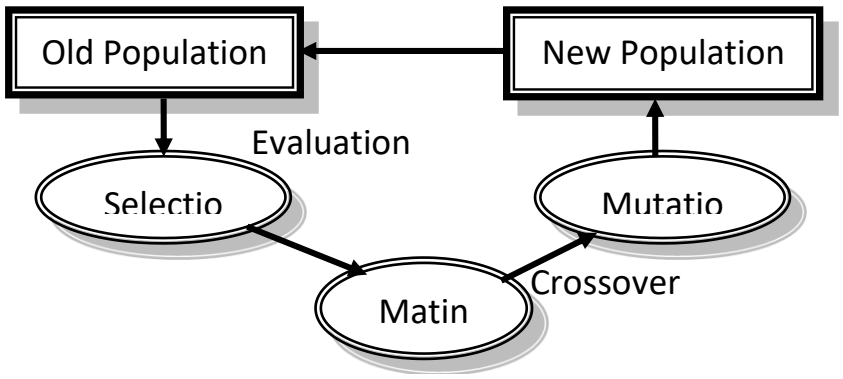

Figure (1) Basic cycle of GA.

5. The following steps described structure of (GA) for the problem $1//(\sum C_i, R_L)$:

Step (1) Initialization

The initial population can be generated at random of *m* individuals (solutions). In this thesis we start with *m* = 20 and 18 from them randomly and the remaining three are given by SPT rule and MST rule.

Step (2) New population

Create a new population by repeating the following substeps until the new population is completed:

a) Selection: In section (3.3) we explained two ways to choose the parent, in our work, we use the first case because it needs less time form the second case, two parent solutions from a current population according to their values (the better fitness, the bigger chance to be selected).

b) Crossover: The crossover operator is the most important operator in GAs. Crossover is a process yielding recombination of bit strings via an exchange of segments between pairs of chromosomes. There are many kinds of crossover, in section (3.4) we introduce some types of crossover and how to work.

c) Mutation: This operater should change from premutation only slightly . The logic of this thought is that asmall amendment in the premutation will give a small curve of it fitness rate only . In this way the information that offered newly by mutation has a rational opportunity to be choose and cross on into future generation . The operation is selecting (and deleting) initialy in GA alter premutation before reinserting this operation at a randomly selected position of the premutation . In the farthest , all priority relation of one operation of the whole other operations are modified . This mutation often dosent alter the value of the fitness and it is has a much same effect , typically .

Step (3) Termination

The procedure stops when a fixed number of generations (or iterations) are execute here (500) iteration. This means that the procedure continued until the population converges to a good, if not optimal, solution to our problem.

6.Computational Experience with GA Algorithm

The execution of problem (P) in the GA alogarithm is compared with 10 problems such the complete enumeration method (CEM).with each n . The instances size are n=4,5,6,7,8,9,10 and the large size are n=11,12,13,14 compare between GA and BAB . [1]. The probles same to that os which it

produced randomly , the p of the processing time , for each job , unformly produced from joint the distribution [1,10]. As well to each job j , an integer due data d ,is creat from joint distribution [(1-TF-RDD/2)TP , (1-TF+RDD/2)TP], the TP is the sum processing time for the all jobs . While lateness agent is the TF , and RDD the close scop of the due data . To the two oarameters RDD and TF the rate 0.2, 0.4, 0.6, 0.8, 1.0 for TF and the values 0.9, 1.0 for RDD ate acount , for each chosen value of n , Tow problems are creat to each of the five ratings of parameters generating 10 problems . Where the problem staged without solutions , with in 1800 second of the time limit . Calculation is ignored for that problem

To clarify that it is not known the optimal pareto front for our problem . In this known for smal l n≤10 only , which by complete enumration method (CEM) is given . The execution of oue alogarithm is then valued to relaize the solutions obtimaed quality by this alogarithm with regard to the solution in (CEM) . All these alogarithm tested by coding them in Matlab 8.3.0 (R2014a) and applied on Intel (R) core (TM) i5 CPU (3210M) @ 2.50 GHZ, with RAM 4.00 GB personal computer.

6.1 Computational Results

Experimental results of proposed GA algorithms and complete enumeration method (CEM) for the problem (P) are given. Table (1), Table (2) and Table(3) show the results of applying algorithms (GA), (CEM) and (BAB) (the algorithm BAB was explained be Abdul-Razaq and Al-shekhly [2]) for the values of $4 \le n \le 7$, $8 \le n \le 10$ and $11 \le n \le 14$ respectively with aim to get the set of approximate solutions (non dominated points) for the problem (P) and minimum sum of $\sum C_i$ and R_L for the problem $\mathbf{1//(\sum C_i + R_L)}$. The results of the set of efficient solutions for (P) are compared with the exact results obtained from (CEM), which generate all solutions for $n \le 10$.

In Table (1),(2) and Table (3) we have:

n: Number of jobs.

Ex: Example number.

Sum: The optimal value for the problem $1//(\sum C_i + R_L)$ obtained by (CEM), or the best value obtained by GA algorithm.

|SE|: Number of efficient solutions obtained by CEM.

$|SE_i|$: Number of efficient solutions, (i=1,2) in $(SE \cap SE_i)$.

Table(1): The performance of the GA algorithms with (CEM) for $4 \le n \le 7$ for the problem (P).

	Efficient Points ($\sum C_i$, R_L) for Problem (P)				
	CEM			**GA**	
n	**Ex**	($\sum C_i$, R_L)	Sum	**($\sum C_i$, R_L)**	**Sum**
4	1	(65,21)	86	**(65,21)**	**86**
	2	(51,18) (45,16)	69	**(51,18)** **(45,16)**	**69**
	3	(61,20) (64,19)	81	**(61,20)** **(64,19)**	**81**
	4	(40,13) (43,12)	53	**(40,13)** **(43,12)**	**53**
	5	(57,14) (61,12)	71	**(57,14)** **(61,12)**	**71**
	6	(52,13)	65	**(52,13)**	**65**
	7	(52,14)	66	**(52,14)**	**66**
	8	(62,18) (67,16)	80	**(62,18)** **(67,16)**	**80**
	9	(32,14) (33,12) (36,11) (40,8)	45	**(32,14)** **(33,12)** **(36,11)** **(40,8)**	**45**
	10	(41,16) (43,13) (46,12)	56	**(41,16)** **(43,13)** **(46,12)**	**56**
5	1	(65,17)	82	**(65,17)**	**82**
	2	(57,15)	72	**(57,15)**	**72**
	3	(70,19)	89	**(70,19)**	**89**
	4	(61,20) (65,7) (70,16)	81	**(61,20)** **(65,7)** **(70,16)**	**81**
	5	(62,20) (65,17) (69,15)	82	**(62,20)** **(65,17)** **(69,15)**	**82**
	6	(87,24)	111	**(87,24)**	**111**
	7	(58,13) (60,12)	71	**(58,13)** **(60,12)**	**71**
	8	(57,18) (58,2)	70	**(57,18)** **(58,2)**	**70**
	9	(46,18) (48,17) (50,15) (58,4)	64	**(46,18)** **(48,17)** **(50,15)** **(58,4)**	**64**
	10	(81,25) (84,21) (87,19) (96,18)	105	**(81,25)** **(84,21)** **(87,19)** **(96,18)**	**105**
	1	(115,29)	144	**(115,29)**	**144**
	2	(73,21) (76,20)	94	**(73,21)** **(76,20)**	**94**
	3	(121,33) (122,32)	154	**(121,33)** **(122,32)**	**154**
	4	(85,4)	104	**(85,4)**	**104**
	5	(85,32)	117	**(85,32)**	**117**

6		(87,30) (91,26) (93,24) (97,23) (106,18) (108,17) (113,16)		**(87,30)** **(91,26)** **(93,24)** **(97,23)** **(106,18)** **(108,17)** **(113,16)**	
	6	(73,20)	93	**(73,20)**	**93**
	7	(76,16)	92	**(76,16)**	**92**
	8	(51,14) (52,12) (57,11)	64	**(51,14)** **(52,12)** **(57,11)**	**64**
	9	(103,29) (105,27)	132	**(103,29)** **(105,27)**	**132**
	10	(106,28) (109,21) (129,20)	130	**(106,28)** **(109,21)** **(129,20)**	**130**
7	1	(117,28) (122,25)	145	**(117,28)** **(122,25)**	**145**
	2	(98,25) (101,23)	123	**(98,25)** **(101,23)**	**123**
	3	(149,36) (151,31) (168,28)	182	**(149,36)** **(151,31)** **(168,28)**	**182**
	4	(85,23) (86,22) (87,20) (88,19) (95,18) (103,17)	107	**(85,23)** **(86,22)** **(87,20)** **(88,19)** **(95,18)** **(103,17)**	**107**
	5	(147,37) (148,34) (154,33) (162,32)	182	**(147,37)** **(148,34)** **(154,33)** **(162,32)**	**182**
	6	(84,19) (87,17)	103	**(84,19)** **(87,17)**	**103**
	7	(167,49) (168,47) (172,41) (182,39) (187,38)	213	**(167,49)** **(168,47)** **(172,41)** **(182,39)** **(187,38)**	**213**
	8	(117,31) (118,29)	147	**(117,31)** **(118,29)**	**147**
	9	(125,38) (126,32) (128,31) (134,30) (138,25)	158	**(125,38)** **(126,32)** **(128,31)** **(134,30)** **(138,25)**	**158**
	10	(83,23) (85,22) (87,20)	106	**(83,23)** **(85,22)** **(87,20)**	**106**

Table (2): The performance of the GA algorithms with (CEM) for $8 \leq n \leq 10$ for the problem (P).

	fficient Points ($\sum C_i$, R_L) for Problem (P)				
	CEM			**GA**	
N	**Ex**	SE	Sum	**SE_1**	**Sum**
8	1	2	187	**2**	**187**
	2	3	162	**3**	**162**
	3	9	236	**9**	**236**
	4	7	158	**7**	**158**
	5	13	140	**13**	**140**
	6	3	200	**3**	**200**
	7	2	201	**2**	**201**
	8	1	135	**1**	**135**
	9	8	137	**8**	**137**
	10	1	78	**1**	**78**
9	1	4	203	**4**	**203**
	2	5	199	**5**	**199**
	3	8	248	**8**	**248**
	4	13	253	**13**	**253**
	5	11	197	**11**	**197**
	6	2	221	**2**	**221**
	7	1	215	**1**	**215**
	8	4	239	**4**	**239**
	9	5	168	**5**	**168**
	10	7	307	**7**	**307**
10	1	4	336	**4**	**336**
	2	7	294	**7**	**294**
	3	13	297	**13**	**297**
	4	16	286	**16**	**286**
	5	17	341	**17**	**341**
	6	2	297	**2**	**297**
	7	2	243	**2**	**243**
	8	4	253	**4**	**253**
	9	11	338	**11**	**338**
	10	11	196	**11**	**196**

Table (3): The performance of the two proposed (BAB and GA) algorithms for $11 \leq n \leq 14$ for the problem (P).

	Efficient Points ($\sum C_i$, R_L) for Problem (P)				
	BAB			**GA**	
n	**Ex**	SE_1	Sum	SE_2	**Sum**
11	1	3	372	**4**	**327**
	2	9	255	**12**	**252**
	3	4	349	**6**	**347**
	4	7	310	**7**	**310**
	5	18	332	**19**	**330**
	6	3	215	**3**	**215**
	7	9	323	**10**	**221**
	8	11	330	**13**	**328**
	9	15	300	**18**	**297**
	10	17	444	**16**	**444**
12	1	3	363	**4**	**362**
	2	4	225	**5**	**224**
	3	9	365	**11**	**363**
	4	11	496	**15**	**493**
	5	13	466	**13**	**467**
	6	2	515	**4**	**513**
	7	6	351	**7**	**351**
	8	2	470	**2**	**470**
	9	8	496	**10**	**494**
	10	15	298	**19**	**294**
13	1	8	473	**9**	**473**
	2	7	435	**8**	**434**
	3	10	363	**13**	**360**
	4	12	368	**13**	**367**
	5	39	570	**39**	**569**
	6	3	307	**5**	**305**
	7	9	436	**11**	**434**
	8	9	382	**11**	**380**
	9	13	374	**15**	**372**
	10	13	304	**14**	**303**
14	1	5	619	**5**	**620**
	2	12	546	**14**	**546**
	3	13	389	**13**	**388**
	4	17	615	**18**	**613**
	5	22	480	**23**	**479**
	6	2	350	**5**	**347**
	7	10	575	**9**	**575**
	8	10	601	**12**	**599**
	9	15	589	**14**	**588**
	10	22	670	**22**	**668**

7. Conclusions

- The GA algorithms was developed to find the set of approximate efficient solutions for the $1//(\sum C_i , R_L)$ problem, and from this set, we can find the optimal or near optimal solution for the problem $1//\sum C_i + R_L$ which is solved for small value of n≤15 by using BAB method .

- From the results of the GA algorithms Tables (1),(2) and (3) show that GA is better than BAB in case of number of efficient solutions.
- From the results of the Table (3) show that GA is better than BAB in case of get the sum if the problem $1//\sum C_i + R_L$.
- Genetic algorithm can be solved $1//(\sum C_i , R_L)$ problem for $n \geq 15$ in reasonable time.
- Note that from the results of the table (3) when n = 14 there is a difference between the results obtained in the number of efficient solutions or in the sum between the two algorithms.

REFERENCES

1. Abdul-Razaq K. F. , "Algorithms for multicriteria scheduling problems" , M. Sc. Thesis, University of AL-Mustansiriyah , college of science, Dept. of Mathematics (2014).
2. Abdul-Razaq T. S and AL Shekhly S.A. Minimizing the Total Completion Times and Range of Lateness on a Single Machine, Transactions on Engineering and Sciences, Volume 4, Issue 3, July-September 2016
3. Alharkan, Ibrahim M. "Algorithms for sequencing and scheduling", Industrial Engineering Department, King Saud University, Riyadh, Saudi Arabia (2005).
4. Anna ŁAWRYNOWICZ, (2011), "GENETIC ALGORITHMS FOR SOLVING SCHEDULING PROBLEMS IN MANUFACTURING SYSTEMS", Foundations of Management, Vol. 3, ISSN 2080-7279.
5. Chen, C.L., Vempati, V.S. and Aljabar, N. (1995), "An application of genetic algorithms for flow shop problems", European Journal of Operational Research, 80, 389-396.
6. Della Croce, F., Tadei, R. and Volta, G. (1995), "A genetic algorithm for the job shop problem", Computers & Operations Research, 22, 25-40.
7. Goldberg, D.E. and Lingle, R. (1985), "Alleles, loci, and the traveling salesman problem", Proceedings of an International Conference on Genetic Algorithms and Their Applications, 154-159.
8. Hoogeveen, J.A .," Invited Review Multicriteria scheduling ", European Journal of Operational Research 167 ,592–623(2005).
9. Murata T, Ishibuchi H, Tanaka H. Multi-objective genetic algorithm and its application to flowshop scheduling. Computers and Industrial Engineering 1996; 30:957-968.
10. Nordström, A.L., and Tufekci, S. (1994), "A genetic algorithm for the talent scheduling problem", Computers & Operations Research, 21, 927-940.
11. Reeves, C.R. (1995), "A genetic algorithm for flow shop sequencing", Computers & Operations Research, 22, 5-13.
12. Smith W.E. , "Various optimizers for single stage production", Naval Research Logistics Quarterly 3/1, 59-66(1956).
13. Syswerda, G. (1991), "Schedule optimization using genetic algorithm", Handbook of Genetic Algorithms, David, L. (Ed.), Van Nostrand Rein-hold, New York, 332-349.
14. Tapan S. , Farhad M. E. , Parthasarati D. "A Branch -and-Bound Approach to the Bicriterion Scheduling Problem Involving Total Flowtime and Range of Lateness ", Management Science, Vol.34(2), 1988, (254-260).

15. Tate, D.M. and Smith, A.E. (1995), "A genetic approach to the quadratic assignment problem", Computers & Operations Research, 22, 73-83
16. Yagiura, M. and Ibaraki, T. (1996), "Genetic and local search algorithms as robust and simple optimization tools", Meta-Heuristics: theory and applications, Osman, I.H. and Kelly, J.P. (Eds.), Kluwer Academic publishers, Boston, 63-82.