A Novel Genetic Algorithm Approach for Solving Flow Shop Problem

Buthainah F Al-Dulaimi[†] and Hamza A.Ali[††]

Faculty of Computer Science and Information Technology,
Al- Isra Private University, Jordan

Summary
Genetic algorithms simulate the survival of the fittest among individuals over consecutive generations for solving a problem. Prior work has shown that genetic algorithms generally do not perform well for shop problems [1]. This paper proposes a solution for the Flow Shop Problem (FSP) or Machine scheduling problem that implements a Genetic Algorithm based solution for Traveling Salesman Problem (GATSP) [2]. The suggested algorithm is stated as a scheduling problem in which a batch of jobs (n-jobs) is to be assigned to a group of machines (m-machine). It aims to maximize the total efficiency of the shop given only the job/machine processing time matrix for machine scheduling.

Key words:
Fflow shop, machine scheduling, genetic algorithms, traveling salesman problem.

1. Introduction

The flow shop sequencing problem (FSSP) with minimum makespan criterion has been treated quite extensively in the past years. The problem of finding the optimal processing sequence is known to be NP-hard [3] and heuristic approaches, as a consequence, have been a favored method for solving the problem when n is large (>50). If part overtaking is not permitted during transfer between machines, the flow shop becomes a permutation flow shop, and the optimal processing schedule is a permutation of the n jobs. Accordingly, search techniques are frequently employed to search the solution space for the permutation of jobs that minimizes the makespan. Search-based algorithms that have been widely used for solving the permutation flow shop have included simulated annealing SA [4-6], taboo search [7&8] and genetic algorithms GA [9-12]. Comparison of performance of SA with GA for flow shop test problems for selected number of jobs and machines concluded that SA out-performed GA in most of the tests where the number of jobs was 50 and less [10]. On the other hand, the solutions provided by GA for the larger problems were mostly superior to those obtained by SA [12]. However, hybrids of GA with local search and with simulated annealing tended to perform better than any one of those algorithms on its own.

Construction heuristics solution, known as NEH algorithm, where a schedule is built iteratively by assigning jobs to a partial schedule are also developed [13]. It is widely acknowledged as one of the best currently available, in terms of solution quality, for minimizing the makespan in a permutation flow shop [12]. The NEH algorithm computes the sum of the processing times for each job and then lists them in non-increasing order of this value. The job at the top of the list is removed and inserted into the partial schedule. The position where it is inserted is determined by considering all the 8 possible positions it can occupy, without altering the relative positions of the jobs already in the partial schedule. The selected position is the one that minimizes the makespan in the partial schedule. This is repeated until the last of the unscheduled jobs is assigned.

Moreover, the flow shop problem is subject to an additional constraint that is no in-process waiting is permitted. Reddy and Ramamoorthy [14] have used Flow Shop, No Intermediate Storage (FSNIS) to denote the latter problem while they used Flow Shop, Infinite Intermediate Storage (FSIIS) to the former problem. A special case of the flow shop is the ordered flow shop, no intermediate storage (OFSNIS) [15&16].

The n job FSNIS problem can be formulated as an n+1 city TSP, while for any number of machines, only permutation sequences need to be considered (in fact all no permutation sequences are infeasible). The special case considers a special class of n job on m machine FSSP with in-process waiting constraint and the criterion of minimum makespan.

This work aims to develop a computer software system for machine scheduling or the flow shop problem using genetic algorithm scheme. The Traveling Salesman Problem solution based on Genetic Algorithm (TSPGA), reported by Al-Dulaimi and Ali [2] is implemented for the required solution.

This paper is organized in the following sequence. After the brief introduction in section 1, genetic algorithms and flow shop problems are defined in sections 2 and 3. Then, section 4 summarizes the main components of Genetic

Manuscript received September 5, 2008.
Manuscript revised September 20, 2008.

algorithm to be adopted in the proposed system. Section 5 outlines in details the suggested Genetic based Flow Shop problem solution. System implementation is described in section 6 and the experimental results are listed and discussed in section 7. Finally, section 8 concludes the paper.

2. Genetic Algorithms

John Holland et. al. [17] developed Genetic Algorithm (GA) as a search algorithm based on the mechanics of natural selection [18] in order to find optimal or near optimal solution. The main idea of GA is that in order for a population of individuals to adapt to some environment, it should behave like a natural system [19]. This means that survival and reproduction of an individual is promoted by the elimination of useless or harmful traits and by rewarding useful behavior. Genetic algorithms are a class of adaptive heuristic search techniques which exploit gathered information to direct the search into regions of better performance within the search space. In terms of time complexity, compared with other optimization techniques (i.e. integer linear programming, branch and bound, tabu search), GA may offer a good approximation for the same big-O time when the state-space is large [20 & 21].

GA belongs to the family of evolutionary algorithms, along with genetic programming, evolution strategies, and evolutionary programming. Evolutionary algorithms can be considered as a broad class of stochastic optimization techniques. An evolutionary algorithm maintains a population of candidate solutions for the problem at hand. The population is then evolved by the iterative application of a set of stochastic operators. These operators usually consist of mutation, recombination and selection or something very similar. Globally satisfactory, if sub-optimal solutions to the problem are found in much the same way as population in nature adapt to their surrounding environment.

An individual degree of adaptation to its environment is the counterpart of the fitness function evaluated on a solution. Similarly, a set of feasible solutions takes the place of a population of organisms. Each individual is a string of binary digits or some other set of symbols drawn from a finite set. It begins with a randomly generated set of individuals (population) then, successive populations, called generations, are derived by applying selection, crossover and mutation operators successively to the previous population. The selection operator chooses two members of the present generation in order to participate in the later operations; crossover and mutation. The crossover operator inter-mixes the alleles of the two parents to produce an offspring. Then mutation operator is activated shortly after crossover, and just like in nature, it exchanges alleles randomly [19].

3. Flow Shop Scheduling Problem

3.1 Description

Flow shop problems are a distinct class of shop scheduling problems [22&23], where n jobs (i = 1, . . . , n) have to be performed on m machines, $< M_1, ..., M_m >$, where $m \geq 1$, as follows. A job consists of m operations; the j^{th} operation of each job must be processed on machine j and has processing time Tij, $1 \leq j \leq n$. A job can start only on machine j if its operation is completed on machine $(j - 1)$ and if machine j is free. The completion time of job i, T_i, is the time when its last operation has completed. Each of n jobs $J_1,..., J_n$ has to be processed on m machines $M_1, ..., M_m$ in that order. Job J_i for i=1... n, thus consists of a sequence of m Process $P_{i1}, ..., P_{im}$; where P_{ik} corresponds to the processing of J_i on machine M_k during an uninterrupted processing time T_{ik}.

This problem is denoted in the literature in $\alpha \mid \beta \mid \gamma$ - notation as $F_m||PC_i$ [24]. Consider an example of flow shop with three machines with the following data, see table 1.

Table 1. Three Machine Flow Shop

Job_i	P_{i1}	P_{i2}	P_{i3}
J_1	1	2	3
J_2	1	2	1
J_3	1	1	1

Figure 1 and Figure 2 show two feasible schedules for the example. Note that in both schedules the order of the jobs differs across machines. For the case (a), we have $T_{max} = 9$ and $\sum T_i = 18$. In case (b), $T_{max} = 8$ and $\sum T_i = 21$. Note that $\sum T_i$ is better than T_{max} in case (i) whereas it is the opposite in case (ii). The example suggests that things very much depend on the objective function.

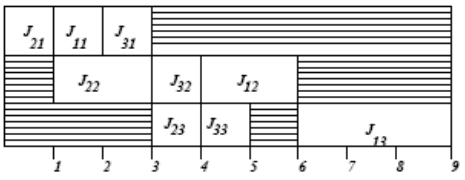


Figure 1. Flow Shop for 3 Machines, case (a)

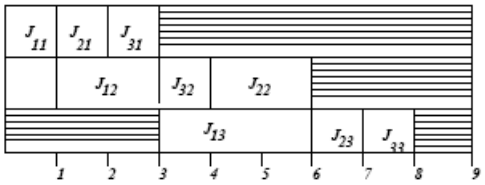


Figure 2. Flow Shop for 3 Machines, case (b)

We look for an efficient solution for the processing order on each M_k such that the time required to complete all jobs is minimized.

3.2 Ordered Flow Shop Problem

A special case of the flow shop is the ordered flow shop, no intermediate storage (OFSNIS).
If the jobs are labeled in ascending, then for any i<k, i & k $\in$ n, $T_{ij} \leq T_{kj}$ for all j $\in$m. with this labeling sequence, 1,. ., n becomes the smallest processing time sequence SPT and sequence n, n-1, . . . , 1 becomes largest processing time LPT sequence. If the smallest processing time for job occurs on the first machine, LPT sequence minimizes the makespan, and vice versa [15].

4. Genetic Algorithms Components:

A detailed description of the genetic algorithm components is cited in reference [2&18] in the course of outlining the TSPGA algorithm which is implemented in the proposed system. An initial sequence population is randomly generated and successive populations called generations are derived by applying the selection, crossover and mutation operators to the previous sequence generation.

4.1 Selection Operations:

A selection operator is applied first. It creates an intermediate population of n "parent" individuals. To produce these "parents", n independent extractions of an individual from the old population are performed [18]. The probability of each individual being extracted should be (linearly) proportional to the fitness of that individual. This means that above average individuals should have more copies in the new population, while below average individuals should have few to no copies present, i.e., a below average individual risks extinction.
The selection operator chooses two members of the present generation to participate in later crossover and mutation operations. Selection process raises the issue of fitness function. A requirement of the selection methods is that the probability f_i of an organism must be the best to be selected.
Two approaches for organism selection were popular; roulette selection and deterministic sampling. The first is based on a probability proportion F_i for each organism calculated by:

$$F_i = f_i \div \sum f_j \quad \ldots\ldots\ldots\ldots (1)$$

While the second is based on determining a criteria C_i for each organism calculated by:

$$C_i = RND\ (F_i * Psize\ ) + 1 \quad \ldots\ldots\ldots\ldots (2)$$

Where RND means rounding to integer and P_{size} means population size.
The selection operator then assures that each organism participates as parent exactly C_i times.
To enhance the selection speed, a newly developed method was reported in [6]. This selection process combines the two mentioned approaches, where relevant organisms with higher fitness are selected and survive.

4.2 Crossover and Mutation Operations:

Once the intermediate population of "parents" (those individuals selected for reproduction or survived) has been produced, the individuals for the next generation will be created through the application of the reproduction operators [18]. These operators involve one or more parents in the operation. An operator that involves just one parent, simulating a sexual reproduction, is called a mutation operator. When more than one parent is involved, sexual reproduction is simulated, and the operator is called recombination. GA uses two reproduction operators - crossover and mutation.

a. Crossover Operations:

To apply a crossover operator, parents are paired together. There are several different types of crossover operators, and each available type depends on what representation is used for the individuals. The one-point crossover means that the parent individuals exchange a random prefix when creating the child individuals. Two-point crossover is an exchange of a random substring, and uniform crossover takes each element in the child arbitrarily from either parent. Order crossover (OX) and partially mapped crossover (PMX) are similar to two-point crossover in that two cut points are selected. For PMX, the selection between the two cut points defines a series of swapping operations to be performed on the second parent [25]. Cycle crossover (CX) satisfies two conditions - every position of the child must retain a value found in the corresponding position of a parent [26], and the child must be a valid permutation. In each cycle, a random parent is selected. CX does not use crossing sites; a cycle is defined in a manner similar to an algebraic permutation group [27].

b. Mutation Operator:

After crossover, each individual has a small chance of mutation. The purpose of the mutation operator is to simulate the effect of transcription errors that can happen with a very low probability when a chromosome is mutated. For example, a standard mutation operator for binary strings is bit inversion. Each bit in an individual has a small chance of mutating into its complement i.e. a '0' would mutate into a '1'.

The Specialized Mutation operator (SMX) is a permutation problem arbitrarily changing single allele value that does not preserve allele uniqueness. The method frequently used for permutation problems is to interchange two randomly selected positions, thus preserving allele uniqueness [15].

5. The Proposed Genetic based Flow Shop (GFS) Problem Solution:

This paper proposes a system for the Flow Shop problem based on Genetic algorithm. This system aims to give an optimal schedule for n jobs on m machines using Genetic Algorithm technique. The block diagram for this system is shown in figure 3 and the algorithm is listed in figure 4. Detailed description of the system is listed in the following sections.

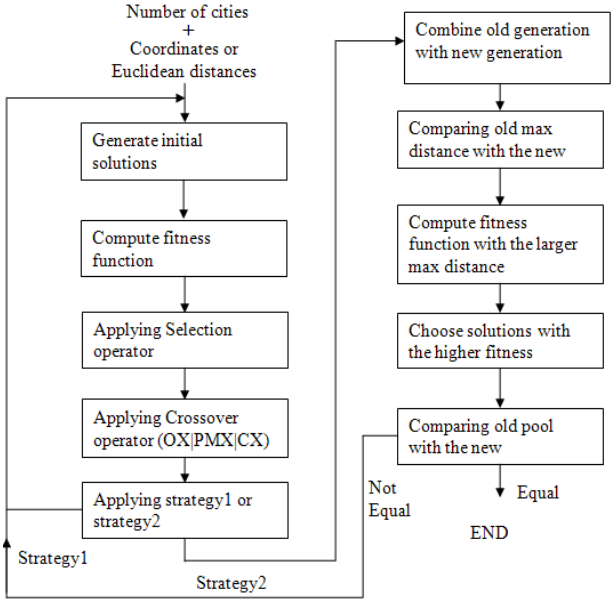


Figure 3. The Flow Diagram of the proposed GFS System.

5.1 Problem Representation and Fitness Function:

The n job Flow Shop, No Intermediate Storage (FSNIS) problem can be represented as an (n+1) city TSP. Several ways exists for representing the equivalent distance matrix for such TSP system, of which Bakers' matrix is used and shown in table 2 [14].

Table 2. The TSP Distance Matrix for FSNIS Problem

0	D_{12}	D_{13}	...	D_{1n}
D_{21}	0	D_{23}	...	D_{2n}
D_{31}	D_{32}	0	...	D_{3n}
...	...	...	...	...
D_{n1}	D_{n2}	D_{n3}	...	0

In the distance matrix of table 2, D_{ik} represents the delay in starting of job k (measured from the start of job i) and the total value of any tour represents the makespan for the corresponding sequence. D_{ik} is calculated by equation 3.

$$D_{ik} = T_{i1} + \max(0, T_{i2} - T_{k1}, T_{i2} + T_{i3} - T_{k1} - T_{k2}, \ldots, \sum_{j=2}^{m} T_{ij} - \sum_{j=1}^{m-1} T_{kj}) \quad \ldots\ldots\ldots (3)$$

(Where T_{ij} denoting the processing time of job i (i=1, ...,n) on machine j (j=1, ..., m).

Now if T_{i1} is subtracted from each row i (≤n) for the distance values in table 2, i.e.

$$D'_{ik} = D_{ik} - T_{i1} \quad \ldots\ldots\ldots (4)$$

The resulting values are listed in table 3.

Table 3. The Reduced Cost Matrix of Table 2

0	D'_{12}	D'_{13}	...	D'_{1n}
D'_{21}	0	D'_{23}	...	D'_{2n}
D'_{31}	D'_{32}	0	...	D'_{3n}
...	...	...	...	...
D'_{n1}	D'_{n2}	D'_{n3}	...	0

The reduced cost matrix values shown in table 3 apply to any of FSNIS as well as any of OFSNIS problems.

The fitness function adopted in this work is computed using equation 5 from the cost matrix

$$f_i = d_{\max} - d_i + d_{last} - d_{first} \quad \ldots\ldots (5)$$

Where last and first in this equation are related to schedule job i, and d_{max} is the maximum time delay.

5.2 GFS Algorithm Operators:

Using the TSPGA system reported by the authors in [2], the optimal machine scheduling is supposed to be obtained. However, in case of optimal schedule does not satisfy the problem constraints, then the proposed GFS system will

be applied. This system works according to the proposed algorithm shown in figure 4.

Algorithm for adjusting optimal schedule for GFS System

```
Begin
input data: distance matrix "D";
input processing time matrix "P";
time = 1 ;  job = 1
for machine = 1 to m
   start operation[job][machine] at time ;
   time = time + P[job][machine]
endfor;
 for job = 2 to n
   for machine = 1 to m
      time  = time[job-1][machine]
      difference = D[job][machine] - P[job][machine]
      if(difference<0) {
            D[job][machine] = D[job][machine] - difference
            D[job+1][machine] = D[job][machine] - difference
            time = time – difference
       else
            time = time  + difference
       endif
       start operation[job][machine] at time ;
       time = time + P[job][machine] ;
   endfor;
  endfor;
end.
```

Figure 4. The Propose Algorithm for GFS System.

In this algorithm the machine schedule is determined by arranging the operations of jobs on machines such that no operation starts until finishing execution of previous machine. The algorithm checks on such criteria in order to handle operation until it finishes execution on previous machine. The execution of any job does not intersect with that of previous job; in this case the algorithm determines the time difference between starting of job operation and ending of previous job operation by applying the equation.

$$\text{difference} = D_{ij} - T_{ij} \quad \ldots\ldots\ldots (6)$$

Where D_{ij} is the delay in starting of job j (measured from the start of job i) and T_{ij} denotes the processing time of job i on machine j).

The algorithm checks each calculated difference value, in case it is positive the job operation starts execution after the (difference time) value from the ending of previous job operation. But in case it is negative then the delay time of current and next jobs will be increased by difference value. Application of this algorithm produces machine schedule that satisfies the problem constraints.

6. System Implementation:

The proposed GFS system is designed to run in two options; default and customized options. It is up to the user to select the option he prefers. Figure 3 illustrated the flow diagram for the proposed default GFS system options. A brief description of each option is included in the following.

6.1 Default System:

When the GFS system runs under this option, the user has no influence on the flow of the system operation. The choice of options is based on the results of experiments. The flow diagram of the default system is displayed on screen to be seen by the user. The user enters only the required data to produce solutions, the steps to enter these data are as follows:

- The system prompts to enter number of jobs and number of machines, (in our prototype design, these numbers must not exceed 26 since letters had been used to represent jobs and machines).
- The system prompts to enter the processing time of each operation of n jobs on m machines, then distance and cost matrices are computed.
- Once distance and cost matrices become available, the system generates initial solutions randomly. Then the GFS cycle continues as in figure 4 till a final solution is obtained.

6.2- Customized System:

When chosen by the user, the system prompts to enter the required data as in default option, then prepares distance matrix and cost matrices. The flow diagram of customized GFS system is shown in figure 4. The user has the following selection choices

- The crossover operator type, i.e. OX, PMX, CX.
- The method of maintaining the mating pool, i.e. choosing one of the following strategies

Strategy1: New generation replaces old generation.
Strategy2: Combination of old and new generations.

Choosing strategy1 have given noisy results and took long implementation time. Therefore, strategy2 was adopted, which determines the longest (maximum) Euclidean distance over old and new generations. This maximum Euclidean distance is then used to compute the fitness of both generations. The resulted solutions were sorted in descending order according to fitness value then the solution with higher fitness was chosen. Finally the GFS system continues the cycle as shown in figure 4.

Implementation of proposed GFS system was on the processing time matrix listed in table 4. Associated

distance and cost matrices for the time matrix of table 4 are listed in tables 5 and 6, respectively.

Table 4. Chosen Processing Time Matrix

Job_i (city)	T_{i1}	T_{i2}	T_{i3}	T_{i4}	T_{i5}
J_1 (a)	5	10	6	6	2
J_2 (b)	6	11	8	6	3
J_3 (c)	7	13	8	8	3
J_4 (d)	9	15	11	10	4

Table 5. The Distance Matrix of Table 4

Job_i (city)	D_{i1}	D_{i2}	D_{i3}	D_{i4}
J_1 (a)	0	9	8	6
J_2 (b)	12	0	10	8
J_3 (c)	15	14	0	11
J_4 (d)	22	18	17	0

Table 6. The Cost Matrix of Table 5

Job_i (city)	C_{i1}	C_{i2}	C_{i3}	C_{i4}
J_1 (a)	0	4	3	1
J_2 (b)	6	0	4	2
J_3 (c)	8	7	0	4
J_4 (d)	13	9	8	0

7. Experiments Discussion of GFS System:

Since strategy1 have given noisy results and took long implementation time, therefore strategy2 is adopted for all system testing. Results for the GFS system, using the three types of crossover (i.e. OX, PMX and CX) with 1% 100% mutation were studied. It is noticed that the schedule depends on a number of parameters, such as the number of jobs, number of machines and number of operations besides processing times and machine loadings.

Considering the results using various processing time matrices, the proposed approach generates optimal and near optimal schedules with respect to the machine imbalance workload. This reflects the robustness of the GA to solve flow shop problems. The best run for "cdba" schedule without using GFS algorithm is shown here in figure 5. It considers for jobs running on five machines. Then, after applying the proposed GFS algorithm, the schedule is recalculated and became as shown in figure 6. From these figures, it is clear that using GFS algorithm have increased the scheduling efficiency. This is manifested by the comparatively less time required to perform all jobs on all machines.

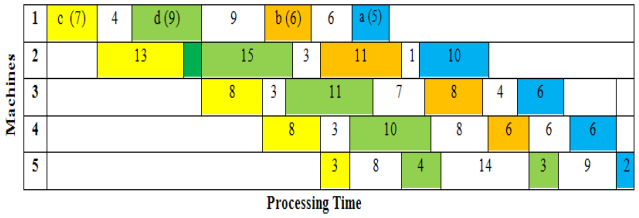


Figure 5. "cdba" schedule before using GFS algorithm

Figure 6. "cdba" schedule after using GFS algorithm

8. Conclusions:

The Flow Shop problem is tackled using Genetic Algorithm based solution for Traveling Salesman Problem (GATSP) in order to obtain optimal machine sequencing schedule. It must be noted that critical path must always pass through machine with large processing times for all jobs. The proposed algorithm targeted the maximization of the total efficiency of the shop given only the job/machine processing time matrix for machine scheduling, with ease and flexibility of data handling and manipulation. The proposed algorithm is necessary to maintain optimal schedule that does not satisfy problem constraint.

References

[1] Matthew Matthew Wall. *A Genetic Algorithm for Resource-Constrained Scheduling*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1996.

[2] Al-Dulaimi B. F. and Ali H. A., "Enhanced Traveling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA)", proceedings of the WASET Conference, 24-26th April, Rome, Italy (2008), PP 70-76.

[3] Rinooy Kan, A.H.G. (1976) Machine scheduling problems: classification, complexity and computations. Martinus Nijhoff, The Hague.

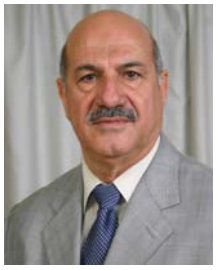
[4] Osman, I.H. & Potts, C.N. (1989). Simulated annealing for permutation flow-shop scheduling. Omega, 17(6), 551-557.

[5] Ogbu, F.A. & Smith, D.K. (1991). Simulated annealing for the permutation flow shop problem. Omega, 19(1), PP 64 - 67.

[6] Ogbu, F.A. & Smith, D.K.. (1990). The application of the simulated annealing algorithm to the solution of the

n/m/C_{max} flow shop problem, Computers &Ops Res., 17(3), 243-253.
[7] Widmer, M. & Hertz, A. (1989). A new heuristic method for the flow shop sequencing problem, Eur. J. Opnl Res., 41, 186-193.
[8] Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem, Eur. J. Opnl Res., 47(1), 65-74.
[9] Syswerda, G. (1991). Scheduling optimization using genetic algorithms, in Handbook of genetic algorithms (edited by L. Davis), 332-349, Van Nostrand Reinhold, New York.
[10] Reeves, C. (1995). A genetic algorithm for flow shop sequencing, Computers Ops Res., 22(1), 5-13.
[11] Chen, C-L., Vempati, V.S. & Aljaber, N. (1995). An application of genetic algorithms for flow shop problems, Eur. J. Opnl Res., 80(2), 389-396.
[12] Murata, T., Ishibuchi, H. & Tanaka, H. (1996). Genetic algorithms for flow shop scheduling problems, Computers Ind. Eng., 30(4), 1061-1071.
[13] Nawaz, M., Enscore Jr., E.E. & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. Omega, 11, 91-95.
[14] Panwalkar S. S. and Woollam C. R., "Flow Shop Scheduling Problems with No In-Process Waiting: A Special Case", Soc. Vol. 30, No. 7, (1979), PP 661-664.
[15] Gonzalez T. and Sahni S., "Flow Shop and Job shop Schedules: Complexity and Approximation", Operation Research, Vol. 26, No. 1, (1978), PP 36-52.
[16] Adiri I. and Amit N., "Open hop and Flow Shop Scheduling to Minimize Sum of Completion Times", Computer and Operation, Vol. 11, No. 3, (1984), PP 275-284.
[17] Holland J. H. et. Al., "Induction: Processes of Inference, Learning, and Discovery", MIT Press, (1989).
[18] Tomassini, M., "Parallel and Distributed Evolutionary Algorithms: A Review. In K. Miettinen, M. M¨akel¨a, P. Neittaanm¨aki and J. Periaux (Eds.)", *Evolutionary Algorithms in Engineering and Computer Science* (pp. 113 - 133). Chichester: J. Wiley and Sons, (1999).
[19] Goldberg D. E., "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley Publishing Company Inc. (1989).
[20] Richard M. Golden. *Mathematical Methods for Neural Network Analysis and Design*. MIT Press, Cambridge, Massachusetts, 1996.
[21] Melanie Mitchell. An Introduction to Genetic Algorithms. The MIT Press, Cambridge, Massachusetts, 1996.
[22] J. Du and J. Y.-T. Leung. Minimizing mean flow time in two-machine open shops and flow shops. Journal of Algorithms, 14:24–44, 1993.
[23] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flow shop and job shop scheduling. Mathematics of operations research, 1:117–129, 1976.
[24] P. Brucker. Scheduling algorithms (Fourth edition). Springer-Verlag, Heidelberg, Germany, 2004.
[25] Oliver I.M., Smith, D.J. and Holland, J.R.C., "A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In John J. Grefenstette (Ed.)", Lawrence Erlbaum Associates. *Proceedings of the 2nd International Conference on Genetic Algorithms, Cambridge, MA, USA, (July 1987)*, (PP 224-230).
[26] Ding C., Cheng Ye. and He M., "Two-Level Genetic Algorithm for Clustered Traveling Salesman Problem with Application in Large-Scale TSPs", Tsinghua Science and Technology, Vol.12, No.4, (2007), PP 459-465.
[27] Phillips D. T., Rarindran A. and Solberg J. J., "Operations Research: Principles and Practice", John Wiley and Sons Inc, (1976).

Buthainah Fahran Al-Dulaimi is an assistance professor at the Faculty of Science and Information Technology at Isra Private University (IPU), Jordan. She got her B.Sc.in 1981 from AL-Mustansryah University/Iraq, M.Sc. in 1999 from University of Baghdad/ Iraq and PhD in 2003 from Institute for Post Graduate Studies in Informatics/ Iraqi Commission for Computers and Informatics (ICCI)/Iraq. Before joining IPU, she worked in the National Computer Center (NCC)/ Iraq as chief programmer/analyzer, executive director of training management and teacher in the same institute. Her research interests include Artificial Intelligence and Software Engineering (model designs).



Hamza Abbass Ali is an associate professor at the Faculty of Science and Information Technology at Isra Private University (IPU), Jordan. He got his B.Sc.in 1968 from Basrah University/Iraq, M.Sc. and Ph.D. in 1973 and 1977 respectively, from The University of London, UK. Before joining IPU, he worked as associate professor at Zarqa Private University (Jordan), visiting professor at University of Aizu (Japan) and associate professor at Basrah University (Iraq). His research interests include Cryptography, Information and Computer Network Security, Artificial Intelligence, Neural Networks and character recognition.