A Genetic Algorithm Approach For Solving Flow Shop Problem

Dr. Hisham Al-Rawi
Faculty of Information Technology
Amman University

Buthayna Fehran
National Computer Center
Baghdad- IRAQ

ABSTRACT

A new approach for solving flow shop problem is presented. The developed and implemented system -based on genetic algorithm- formulates the flow shop problem as a traveling salesman problem and solves the machine-job scheduling problem accordingly. Special selection, crossover, and mutation operators are used within the genetic system. Results showed fast convergence toward optimal schedule.

Keywords: Scheduling; Flow Shop Problem; Genetic Algorithm.

ملخص

توجه الخوارزمية لحل مشكلة إنسياب العمل

تعرض الورقة توجها جديدا لحل مشكلة انسيابية العمل الورشي. يعتمد النظام الحاسوبي المعروض على الخوارزمية الجينية وعلى تمثيل مشكلة إنسياب العمل الورشي على صورة مشكلة البائع المتجول. وعلى هذا الأساس تعرض الورقة حل مشكلة جدولة الماكنة ـ العمل. تتطرق الورقة الى الاختيارات المناسبة للعوامل الأساسية في الخوارزمية الجينية. أظهرت النتائج المعطاة إنحيازا شديدا نحو الجدولة المثلى.

تاريخ تسلم البحث: 2001/7/11 تاريخ قبول البحث: 2001/10/10

1. Introduction

The Flow Shop Problem may be stated as a scheduling problem in which n jobs have to be processed on m machines[1]. In a machine shop a batch of jobs is to be assigned to a group of machines in a way to maximize the total efficiency of the shop.

Flow shops are ordered sets of m processors (or machines), $<P_1, ..., P_m>$, $m \geq 1$, the processing time required by task j of job i is denoted by P_{ij}, $1 \leq j \leq m$. For any job i, task j, is performed on processor P_j. For any job i the processing of task j, $j \geq 2$ can begin only after task j-1 has been completed. The general flow shop problem can be formulated as follows. 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, i=1, ..., n, thus consists of a sequence of m operations $O_{i1}, ..., O_{im}$; O_{ik} correspond to the processing of J_i on M_k during an uninterrupted processing time P_{ik}. M_k, k=1, ..., m can handle at most one job at a time. It is to find a processing order on each M_k such that the time required to complete all jobs is minimized.

Flow shop problem can be classified into two categories depending on the availability (or the requirement) of intermediate storage. These categories are; the FSNIS (Flow Shop, No Intermediate Storage) and the FSIIS (Flow Shop, Infinite Intermediate Storage). The former is more complicated due to the additional constraint of the storage availability. The FSNIS is the concern of this paper.

Flow shop problem can be classified into two categories depending on the availability (or the requirement) of intermediate storage. These categories are; the FSNIS (Flow Shop, No Intermediate Storage) and the FSIIS (Flow Shop, Infinite Intermediate Storage). The former is more complicated due to the additional constraint of the storage availability. The FSNIS is the concern of this paper.

The work presents a solution to the flow shop problem with no intermediate storage (FSNIS) based on genetic algorithm. A Genetic algorithm Flow Shop (GFS) scheduler system is presented. The system deals with the problem of FSNIS as a traveling salesman problem. The presented GFS system is an extension to a previously reported genetic algorithm system [6] for solving the traveling salesman problem (TSP).

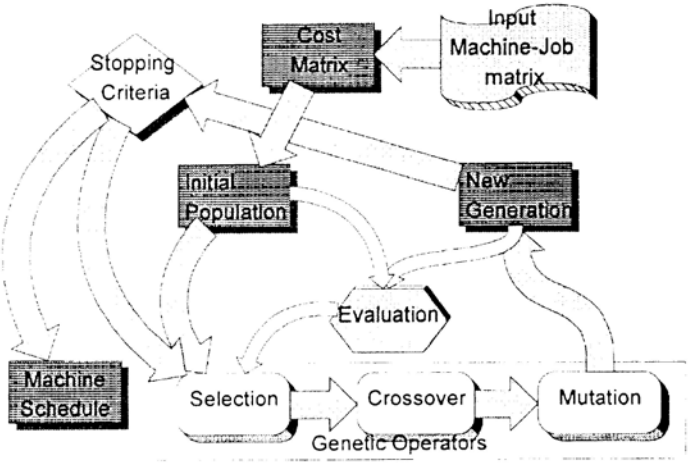

Figure 1 Genetic Algorithm Flow Shop System.

2. *Genetic Algorithm*

The so-called Genetic Algorithms(GAs) are new algorithms than can attack difficult (NP-Complete) computation problems such as TSP[6,7]. These algorithms developed by John Holland at the University of Michigan in the mid-1970s. As the name implies, it is an iterative procedure based on various biological principles. GA encodes information into strings just as living organisms encode characteristics into strands of DNA[8]. A string in GA is analogous to a chromosome in biology, chromosomes are composed of genes which take some values called alleles, strings are composed of features which may take different values, one or more chromosomes are combined to form genotype, combined strings are called a structure.

In natural systems, the organisms formed by interaction of the genotype with its environment are called phenotype, in artificial genetic, the structures decode to form a parameter set. As with biological parameters, two strings combine and contribute part of their characteristics to create their child. This child joins the pool and fight to produce the next generation[9].

2.1 Genetic Algorithm Cycle

A GA is one of search algorithms that use probability to guide their search with no constraints within the search space (like the existence of derivatives)[10], its components are[7] :

- An initial population of random chromosomes, coded in a relevant form.
- An objective function to measure the near of each chromosomes to the solution of the problem.
- A selection operation to select chromosomes based on their objective function values, the better ones are with high probability of contributing one or more offspring.
- A crossover operation which takes pair of selected chromosomes and randomly combines elements of each to produce offspring.
- A mutation operation which changes offspring alleles randomly with a low probability.

2.2 The Differences Between Genetic And Conventional Algorithms

GAs differentiate from conventional algorithms in the following points[9]:

- GAs work with parameters coding. To solve the problem with GAs, the first step is to code the parameter as a finite length string, while traditional algorithms work on the parameters themselves.
- GAs search from a population of points simultaneously . The probability of finding a false peak is reduced because of climbing many peaks in parallel, while traditional algorithms move from a single point in the search space to the next using some transition rule to determine the next point which may locate false peak in multimodal spaces.
- GAs use objective function values associated with individual strings, not derivative or other auxiliary knowledge but traditional methods need derivatives like in gradient techniques to climb the current peak, and require auxiliary knowledge like in greedy techniques to access to most if not all tabular parameters.
- GAs use probabilistic transition rules, GAs use random choice to guide a search toward improved regions of search while traditional algorithms use deterministic rules.

3. Genetic Algorithm Flow Shop System

The developed Genetic algorithm Flow Shop (GFS) system consists of the stages is shown in figure 1. The machine sequencing problem is represented in the early stages of the system as strings or organisms. Each organism may represent a sequence of the alphabet indicating machine sequencing. The following subsections describe the different stages of the system. System input is a matrix representing the processing time of each job on each machine.

An initial tours population is randomly generated and successive sequence populations called generations are derived by applying the selection, crossover and mutation operators to the previous sequence generation. As shown in figure 1 a current generation being acted upon by the three operators to produce the successive generation.

3.1 Problem Formulation and Fitness Function

The n job FSNIS problem can be formulated as an (n+1) city traveling salesman problem. There are several ways of representing the equivalent distance matrix for the TSP. Bakers matrix [4] is one of these, and it is shown in table 1. In the distance matrix, 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. Using the following equation [4] to compute D_{ik}

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

where P_{ij} denotes the processing time of job i(i=1, ...,n) on machine j(j=1, ..., m)

Subtracting P_{i1} from each row i ($\leq$n) in table 1 reduces the matrix of table 1 to table 2 as in the following equation

$$D'_{ik} = D_{ik} - P_{i1} \quad \ldots(2)$$

The reduced matrix applies to any FSNIS as well as any open flow shop no intermediate storage (OFSNIS) problem.

The fitness function used here computed by using the following equation and the cost matrix

$$f_i = d_{max} - d_i + d_{last} - d_{first} \quad \ldots(3)$$

where last and first are related to schedule i.

3.2 Selection Operation:

The selection operator chooses two members of the present generation to participate in the later operations; crossover and mutation. There are two popular approaches for implementing selection. The first is the roulette selection, and the second is the deterministic sampling. System performance using each of the above two selection methods was poor. Convergence (measured in time units or number of generations) is found slow. The basket selection method suggested in [6] is adopted. The newly developed method is a modified combination of the above two methods. The fitness, is calculated relatively within each generation

3.3 Crossover And Mutation Operations:

Problems such as those involving the optimization of a certain order of parameters, are naturally coded permutation organisms such as (i, b,g,...,e). The use of crossover and mutation operators on these is more subtle, such as (i, b, g, ..., b). Therefore such problems involve special crossover and mutation operators. Two point crossover operator would have to be modified to work with such problems. Exchanging parts of two solutions will usually result an invalid solution.

Three types of special crossover operators reported for permutation problems are selected to be examined and used in the GFS system. These are: order crossover (OX), partially matched crossover (PMX), and cycle crossover (CX). One child from a pair of parents is considered [6]. Arbitrarily changing single allele value would not preserve allele uniqueness. The mutation method used in GFS system is the swap mutation; that is to interchange two randomly selected positions, thus preserving allele uniqueness.

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

Table 1: 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

Table 2: The Cost Matrix Of Table 1

```
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
```

Algorithm 1: Adjusting Critical Path Schedule

3.4 Replacement

The weak parents replacement is implemented in the GFS system. In this method both parents and children are combined together and a number (equal to the population size) of the fittest individuals is selected to represent the new generation.

3.5 Critical Path Adjustment

It was found from the different flow shop problems solved by the system that the optimal schedule obtained by the system may have some problems in the critical path. An overlap in processing times was obtained in certain cases. The GFS system ends with a population of candidate solutions for the optimal job sequencing. Generally the solution with the higher fitness is selected and the job-machine schedule is worked out according to that squence. If a processing time overlap problem found in the critical path then it is suggested to select the second candidate (sequence) solution. An algorithm for detecting any such overlap problem and adjust that problem by selecting the second fittest solution can developed. Obviously, selecting the second candidate solution means accepting the solution which is not the best in terms of distance, cost or processing time.

It is worth to mention that even the second candidate solution may not necessarily overcome the overlapping problem. Once the second candidate solution is less fit then the total cost is higher than the first fittest solution, and accordingly there may not be any problem in the critical path, but that is not guaranteed.

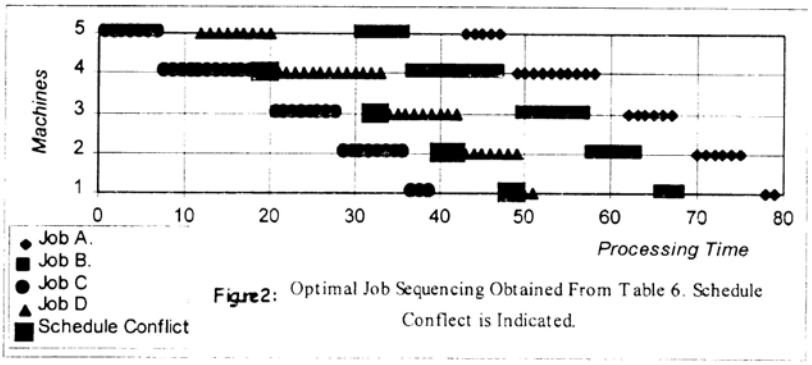

Figure2: Optimal Job Sequencing Obtained From Table 6. Schedule Conflect is Indicated.

However, it was found that even with the first candidate solution it is possible to solve this overlapping problem. An algorithm was developed to detect such overlaps and then determines the difference time between the starting of job operation and the end of previous job operation by applying the following equation

$$difference = D_{ij} - P_{ij} \qquad ...(4)$$

where D_{ij} represents the delay in starting of job j (measured from the start of job i), and P_{ij} denotes the processing time of job i on machine j.

The schedule adjusting algorithm checks difference value, in case it is positive then the job operation starts execution after the difference time value from the end of previous job operation, and in case it is negative then the delay time of current and next jobs is increased by the difference value as shown in algorithm 1. The schedule adjusting algorithm keeps the same optimal job sequencing obtained, deals with the critical path schedule by advancing or delaying the specified job schedule by the difference value calculated, and finds out the optimal -nonconflicting- job-machine schedule based on the optimal job sequencing obtained.

	M1	M2	M3	M4	M5
Ja	5	10	6	6	2
Jb	6	11	8	6	3
Jc	7	13	8	8	3
Jd	9	15	11	10	4

Table 3: System Input; Processing Time Matrix

	Ja	Jb	Jc	Jd
Ja	0	9	8	6
Jb	12	0	10	8
Jc	15	14	0	11
Jd	22	18	17	0

Table 4: Distance Matrix Of The FSNIS Problem of Table 3

	Ja	Jb	Jc	Jd
Ja	0	4	3	1
Jb	6	0	4	2
Jc	8	7	0	4
Jd	13	9	8	0

Table 5: Cost Matrix Of The FSNIS Problem of Table 3

machine has to be ready to start processing job i directly after coming out of machine j. Table 3 shows an imbalance machine workload.

After entering the data of table 3 to the GFS system the system starts formulating the problem as a TSP problem. Table 4 and table 5 give the distance and cost matrices calculated by the system. Table 6 shows the results obtained in just a single generation using a population size of 6, 0.1 mutation rate and the three considered types of crossover operators. The system using OX, PMX, and CX crossover operations converged toward the solution within just single generation. The results using each of the crossover operations are identical. The total Euclidean distance of each schedule and the relative fitness among the population members are also given. The result obtained from this stage represents the optimal job sequencing on each machine.

Figure 2 shows a plot of the sequence as a schedule of the four jobs on the five machines according to the fittest (sequence) solution obtained from table 6. No conflict in Machine Number five job schedule. The critical path can be seen on machine number 4. This machine suffers from schedule overlaps between job C and job D. This overlap caused schedule conflict in the remaining machines schedules, as can be seen from figure.

Applying the schedule adjustment algorithm permits the use of the same job sequence of the fittest solution obtained. Figure 3 shows the optimal flow shop schedule obtained with no intermediate storage.

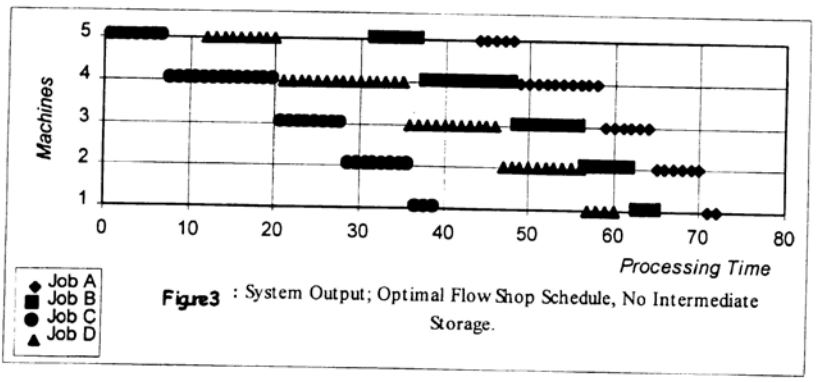

Figure3 : System Output; Optimal Flow Shop Schedule, No Intermediate Storage.

5. Conclusion

A new approach for solving flow ship problem using genetic algorithm was presented. A Genetic algorithm Flow Shop (GFS) scheduler system was implemented. The system consists of ; (1) a stage for formulating the problem as a TSP problem, calculate a distance and a cost matrix for the given problem, (2) a stage for generating randomly a population of machine sequences, (3) a genetic algorithm subsystem consisting of (basket) selection, (OX, PMX, and CX) crossover and (swap) mutation operations with a guided search based on a fitness function reported earlier, and (4) a stage for detecting and adjusting any schedule conflicts due to the critical path schedule overlaps. System output is a job-machine processing schedule as that of figure 3.

System performance shows that genetic algorithm is fast in converging toward an optimal solution for such NP-complete problem. The stopping criteria considered in the system is either a three time solution repetition in successive generations or when no improvement in the solution obtained in successive generations is found. The effect of selecting any of the special crossover operators is very minor in small scale problems such as the one reported in this method for solving other machine scheduling problems.

OX Operator				
Schedule	d	f	F	C
bdca	18	10	0.25	3
cdba	19	9	0.22	3
dbca	21	7	0.17	3
bcda	21	7	0.17	3
dcba	21	7	0.17	3
abcd	28	0	0.00	1
PMX Operator				
Schedule	d	f	F	C
bdca	18	10	0.25	3
cdba	19	9	0.22	3
dbca	21	7	0.17	3
bcda	21	7	0.17	3
dcba	21	7	0.17	3
abcd	28	0	0.00	1
CX Operator				
Schedule	d	f	F	C
bdca	18	10	0.25	3
cdba	19	9	0.22	3
dbca	21	7	0.17	3
bcda	21	7	0.17	3
dcba	21	7	0.17	3
abcd	28	0	0.00	1

Table 6: First Genetic Algorithm Generation Using OX, PMX, and CX crossover operators; Six candidate solutions

d	Euclidean distance
f	Fitness Value
F	Roulette Wheel probability
C	Deterministic Sampling probability

References

1. Phillips D. T., Rarindran A., and Solberg J. J.; ***Operations Research: Principles And Practice***, John Wiely, 1976.
2. Gonzalez T., and Sahni S.; ***Flow Shop And Job Shop Schedules: Complexity And Approximation*** , Operations Research, Vol. 26, No. 1, pp 36-52, 1978.
3. Lageweg B. J., Lenstra J. K., and Rinnooy Kan A. H. G.; ***A General Bounding Scheme For The Permutation Flow Shop Problem***, Operations Research, Vol. 26, No. 1, pp 53-67, 1978.
4. Panwalkar S. S., and Woollam C. R.; ***Flow Shop Scheduling Problems With No In-Process Waiting: A Special Case***, Journal Of Operation Research Society. Vol. 30, No. 7, pp 661-664, 1979.
5. Adiri I., Amit N.; ***Open Shop And Flow Shop Scheduling To Minimize Sum Of Completion Times***, Comput. And Ops. Vol. 11, No. 3, pp 275-284, 1984.
6. Al-Rawi H., and Fehran B.; ***A Genetic Algorithm Approach For Solving The Traveling Salesman Problem*** , Proceeding of ACIDCA 2000 Monisteir; Tunissia, Vol IM pp99-103, 2000.
7. Matthews R. A. J.; ***The Use Of Genetic Algorithms In Cryptanalysis***, Cryptologia, Volume XVII Number 2, pp 187-201, April 1993.
8. Grant K.; ***An Introduction To Genetic Algorithms***, C/C++ Users Journal, pp 45-57, March, 1995.
9. Goldberg D. E.; ***Genetic Algorithms In Search, Optimization, And Machine Learning***, Addison Wesley, 1989.
10. Grefenstette J. J.; ***Optimization Of Control Parameters For Genetic Algorithms***, in ***Genetic Algorithm*** Ed. Buckles B.P. and Petry F. E., IEEE Computer Society Press, pp 5-11, 1992.