Solving Job-Shop Scheduling Problem Using Genetic Algorithm Approach

Wathiq N. Abdullah
University of Baghdad
College of Education- Ibn Alhaitham

Abstract

An effective job shop scheduling (JSS) in the manufacturing industry is helpful to meet the production demand and reduce the production cost, and to improve the ability to compete in the ever increasing volatile market demanding multiple products.

In so many combinatorial optimization problems, job shop scheduling problems have earned a reputation for being difficult to solve. Job-shop scheduling is essentially an ordering problem. A new encoding scheme for a classic job-shop scheduling problem is presented. The aim is to find an allocation for each job and to define the sequence of jobs on each machine so that the resulting schedule has a minimal completion time. Genetic algorithm that has demonstrated considerable success in providing efficient solutions to many non polynomial-hard optimization problems is used to solve job-shop scheduling problem. The schedules given by genetic algorithms are constructed using a priority rule and under several constraints. After a schedule is obtained a checking operation is applied to ensure that the solution is feasible. The approach is tested on a set of instances. The results validate the effectiveness of the algorithm.

Introduction

Scheduling problems are usually approached with a combination of search techniques and heuristics. Scheduling belongs to NP-complete problems. Such problems are likely to be unmanageable and cannot be solved by combinatorial search techniques. Moreover, heuristics alone cannot guarantee the best solution. They involve a competition for limited resources; as a result, they are complicated by many constraints [1].

Manufacturing environments in the real world are subject to many sources of change which are typically treated as random occurrences, such as new job releases, machine breakdowns, etc. In most manufacturing environments, there is a requirement to use the available resources as

efficiently as possible [2]. Due to their dynamic nature, scheduling problems are rather computationally complex -- i.e., the time required to compute an optimal schedule increases exponentially with the size of the problem. [3] They are complex tasks that can be formulated using a constraint-based representation.[4]

Reasons for scheduling complexity include [5]: (*i*) Scheduling is a feasibility problem. The final solution must accomplish all the problem constraints. Another objective to be satisfied is the optimization of an evaluation function, adjusting to certain criteria as cost, lateness, process time, inventory time, etc. (*ii*) Some scheduling problems have many constraints due to the unavailability of resources, due dates, etc. [4]

Scheduling problem involves allocating limited resources to operations (activities) over time. [4] In order to complete a schedule, all the operations must be completed. [2] Job shop scheduling problem is one type of the scheduling problems.[6] ((that)) is a very important practical problem. [7] It is among the hardest combinatorial optimization problems[8].

Efficient methods of solving it can have major effects on profitability product quality. The problem is to minimize the total elapsed time between the beginning of the first operation and the completion of the last operation (the makespan). Other measures of scheduling quality exist, but shortest makespan is the simplest and most widely used criterion. In general, the difficulty of the Job-Shop Scheduling problem makes it very hard for conventional search-based methods to find near-optima in reasonable time. This has led to recent interest in using genetic algorithms to address these problems. [7]

The remainder of this paper is organized as follows: the job-shop scheduling problem, the design of the system of solving job-shop scheduling problem using genetic algorithm including the representation of the chromosome, and finally, the results of the system and some concluding remarks are given.

Job- Shop Scheduling Problem using Genetic Algorithm

Since job shop scheduling is actually a constrained multi-sequencing problem, a lot of research concentrates on the problem representation by encoding operation sequences for the machines [9]. A schedule is defined by a complete and feasible ordering of operations to be processed on each machine [10].

The JSP is a scheduling problem that considers *M* different machines and *N* different jobs. Each of the jobs consists of *Q* operations and each of

the operations requires a different machine. All the operations of each job are processed in a fixed processing order [11] which specifies the precedence restrictions [3]. The operations of a job are totally ordered so that no operation of a job can start before the completion of its predecessor [12]. Two operations cannot be scheduled at the same time if they both require the same machine [10]. Each operation is characterized by the required machine and the processing time[11]. This time is known a job arrive at the shop all jobs are ready to start at time zero [3]. A job is processed on one machine at a time. Machines are available continuously [11] and there are no machine breakdowns [10].

Scheduling systems typically rely on priority rules, which have therefore become the subject of intense study [13]. A schedule builder produces a single schedule on the basis of a certain priority rule [14]. A priority rule is used to select the operation to be scheduled at each step of the algorithm, among the set of unscheduled operations. In some cases, the operations are selected according to a predetermined order, which is equivalent to assign to each operation a unique priority value. The operation with the smallest priority value has the highest priority. This priority value can be computed in a static way (it remains constant throughout the process) [12].

Two kinds of constraints need to be considered for the Job-Shop Problem [8] as follows:

(*i*) Operation precedence constraint for a given job:

As usual, let *cjk* denote the completion time of job *j* on machine *k* and let *tjk* denote the processing time of job *j* on machine *k*. For a job *i*, if the processing on machine *h* precedes that on machine *k*, we need the following constraint:

$$cik - tik >= cih \quad \ldots\ldots (1)$$

(*ii*) Operation un-overlapping constraint for a given machine:

For two jobs *i* and *j*, both need to be processed on machine *k*. If job *i* comes before job *j*, we need the following constraint [11]:

$$cjk - cik >= tjk \quad \ldots\ldots (2)$$

The goal of the Job-Shop Problem is to choose for each operation a suitable machine and a starting time so that the maximum completion time Cmax (the makespan) is minimized. [15, 16, 17]

The genetic algorithm (GA) that has received a rapidly growing interest in the combinatorial optimization community and has shown great power with very promising results from experimentation and practice of

many engineering areas [11] has been used with increasing frequency in the context of job shop scheduling problems [3].

Encoding a Schedule and Representation of a Chromosome

As we mentioned earlier, the job-shop scheduling problem is a sequencing problem of *M* machines and *N* jobs. Each job consists of a chain of operations. All the operations of each job are processed in a fixed processing order.

Suppose that the available set of total operations of *N* jobs be *OP*, the available set of machines is *M*, and the indices of the operation *op* be *i* and *k*, then we have:

$OP = \{op_{ik} \mid i = 1, 2, \ldots, N \text{ and } k = 1, 2, \ldots, M\}$ …….. (3)

Where *i* is the job number, and *k* indicates the operation number of job*i*.

The *ik* indices of the operation *op* are encoded as a sequence number (*index*). For example, consider the problem with two jobs, and each job has three operations. For this problem, the *OP*= (*op*11, *op*12, *op*13, *op*21, *op*22, *op*23). Note that the *op*11, *op*12, and *op*13 are elements of job 1, the *op*21 and *op*22 are of job 2. Assign a number, from 1 to 6, to each operation in the order of job indices, we have, 1 = 11, 2 = 12, 3 = 13, 4 = 21, 5=22, and 6 = 31, as shown in Table 1. This sequence number is used as an index of a chromosome's gene.

The chromosome's gene can be represented by two fields: the first field (*Mach*) consists of *M*-bit string (*M* is the number of machines), where each bit corresponds to one machine. If the operation is to be processed on a particular machine, the corresponding bit assumes value (1), otherwise it is (0). The second field of a gene is the completion time of the operation (*C*). A sample of a chromosome is shown in Figure 1.

Generation of Initial Population

Initialization is the process of generating a new sequence of chromosomes. The initialization procedure produces *pop_size* chromosomes- where *pop_size* denotes the population size- by setting (1) at random position in each (*mach*) field of a chromosome and filling the remainder bits of the field by (0's) until the whole chromosome is filled and repeat the same procedure until the initial population is filled.

Evaluation of Fitness

The evaluation of a chromosome's fitness involves finding the total completion time (*TCT*) of the operations on each machine. The total completion time on each machine can be calculated by summing the time required to process each operation on a certain machine. The total completion time having the maximum value is the fitness value of the

chromosome (i.e., the fitness value is the makespan). The formula of fitness function is as follows [1]:

Fitness of a chromosome = $\underset{1\le k\le M}{Max}\{TCT_k\}$ …….. (4)

Where:

k is the machine number,

M is the number of machines, and

$$TCT_k = \left(\sum_{i=1}^{n} Op_i Time \times Chrom[Op_i[bit_k]]\right) \times pen \quad \text{…….. (5)}$$

Where:

n is the total number of operations,

$Op_i Time$ is time required to perform the operation number i as given in table 1,

$Chrom\ [Op_i\ [\ bit_k\ ]]$ is either (0) or (1) as exist in the chromosome at operation number (i) at bit number (k), and

pen is the penalty variable that is (1) for legal schedule and (100) for illegal one.

A randomly built initial population and new populations generated in advance may contain illegal schedules. Illegal schedule is a schedule that violates the problem constraints. In order to avoid the selection of illegal schedule a penalty is forced on this schedule. The high penalty (i.e. 100) decreases the probability that the violators will not procreate.

Evaluation of Best-Fit Chromosome

After finding the fitness values of the chromosomes, we will evaluate the best fit chromosome. The best chromosome in a population to be selected is the one that has minimum fitness value among the population finesses, i.e., for which the makespan is minimum.

Construction of GA Operators

The main genetic operators (Selection, Crossover, and mutation) are applied iteratively on each population. The selection operator selects chromosomes for crossover according to their fitness. Selection operator adopted in this system uses tournament selection approach. The crossover operator cuts the parent chromosomes at the randomly selected point denoted by the vertical line and exchanging parental genes after the cut to create new chromosomes. Figure 2(a) demonstrates the crossover operator.

The mutation operator randomly selects a gene in a chromosome and replaces it by a gene randomly selected from the pool. In the example

shown in Figure 2(b), the chromosome is mutated in its second gene, which is replaced by the gene [0 0 1 0] chosen from the pool of genes.
Figure 3 shows the detailed steps of the system of job-shop scheduling problem. The system was implemented using Visual Basic 6.0 language.

Experimental Results

Different cases on which the system was run. These cases are taken from different perspectives: population size, probability of genetic algorithm operators, and maximum number of generations. The parameters values that used with each instance are given in Table2.

Various problem instances are shown in Table 3 through Table 8. These instances differ from each other by the number of machines in the system, the number of jobs, and the number of operations required to be processed in each job. The time required to perform each operation of a job is generated randomly.

The *(M, T)* field in instances tables means that the operation of the job must go to machine (*M*) for (*T*) units of time. At the end of each instance table the resultant makespan from that instance is given.

Conclusions

This paper presents a genetic algorithm for Job Shop Scheduling Problem. JSSP uses direct representation in which a chromosome represents a schedule. The schedules are constructed using a priority rule.

Various problem instances are tested to show the results of the system. These instances are differ from each other by the number of jobs, the number of operations required to be processed in each job, the number of machines, and the values of genetic parameters. The experimental results show that the algorithm produced optimal or near-optimal solutions on all instances. The schedules obtained are presented in the form of tables.

The system contains a number of machines. Each machine can process only one machine at a time. Each job consists of sequence of operations in a predefined precedence order. The problem is to find a schedule having a minimum total time often called the *makespan* of a schedule.

Four genetic parameters including: the population size, the crossover rate, the mutation rate, and the stopping condition are used as the problem parameters.

References

[1] Negenvitsky, M., Artificial Intelligence: A Guide to Intelligent Systems. 2nd edn, Addison-Wesley, Harlow, England, 2005.

[2] Cheeseman, M., *The Application of Naturally-Inspired Algorithms to the Job-Shop Scheduling Problem*, Rolls-Royce Strategic Research Centre, Sinfin A -28,Rolls-Royce plc.

[3] Lin, S.-C., Goodman, E.D., and Punch, W.F., *A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems*, Technical Report GARAGe97-02-08, Genetic Algorithms Research and Applications Group, Michigan State University, 1997.

[4] Garrido, A., Salido, M. A., Barber, F., and Lopez, M. A., *Heuristic Methods for Solving Job-Shop Scheduling Problems*,Spain, 1998.

[5] Fox, M. S. and Sadeh, N., Why is Scheduling difficult? A CSP Perspective, In 9th European Conference on Artificial Intelligence, 1990.

[6] Coello, C. A., Rivera, D. C., and Cortes, N. C., Use of an Artificial Immune System for Job Shop Scheduling, Evolutionary Computation Group, Av. Institute Politecnico National No. 2508, Col. San Pedro Zacatenco, Mexico, 2003.

[7] Fang, H.-L., Ross, P.,and Corne, D., A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems, Proceedings of the Fifth International Conference on Genetic Algorithms, S. Forrest (ed.), San Mateo: Morgan Kaufmann, pages 375-382, 1993.

[8] Gen, M., and Cheng, R., Genetic algorithms and engineering design. Wiley, New York, 1997.

[9] Cheng, R., Gen, M., and Tsujimura, Y., *A tutorial survey of job-shop scheduling problems using genetic algorithms part II: Hybrid genetic search strategies*. Computers and Industrial Engineering 36, 343–364, 1999.

[10] Thamilselvan , R., and Balasubramanie, P., *Integrating Genetic Algorithm, Tabu Search Approach for Job Shop Scheduling*, (IJCSIS) International Journal of Computer Science and Information Security, Vol. 2, No. 1, 2009.

[11] Liu, T.-K., Tsai, J.-T., and Chou, J.-H., *Improved genetic algorithm for the job-shop scheduling problem*, Taiwan, R.O.C., 2005.

[12] Artigues, C., Lopez, P., and Ayache, P.-D., *Schedule generation schemes for the job*-shop problem with sequence-dependent *setup times: dominance properties and computational analysis*, 2003.

[13] Haupt, R., A survey of priority rule-based scheduling, OR Spektrum 11, 3–16, 1989.

[14] Mattfeld, D. C., and Bierwirth, C., *An efficient genetic algorithm for job shop scheduling with tardiness objectives*, European Journal of Operational Research 155, pages 616–630, San Francisco, CA, 2004.

[15] Hmida, A. B., Huguet, M.-J., Lopez, P., and Haouari, M., *Climbing Depth-bounded Discrepancy Search for Solving Flexible Job Shop Scheduling Problems*, 2007.

[16] Ombuki, B. M., and Ventresca, M., *Local Search Genetic Algorithms for the Job Shop Scheduling Problem*, Brock University, Department of Computer Science, Canada, 2002.

[17] Wang, Y. M., Yin, H. L., and Wang, J., *Genetic algorithm with new encoding scheme for job shop scheduling*, Springer-Verlag, London, 2009.

Figures

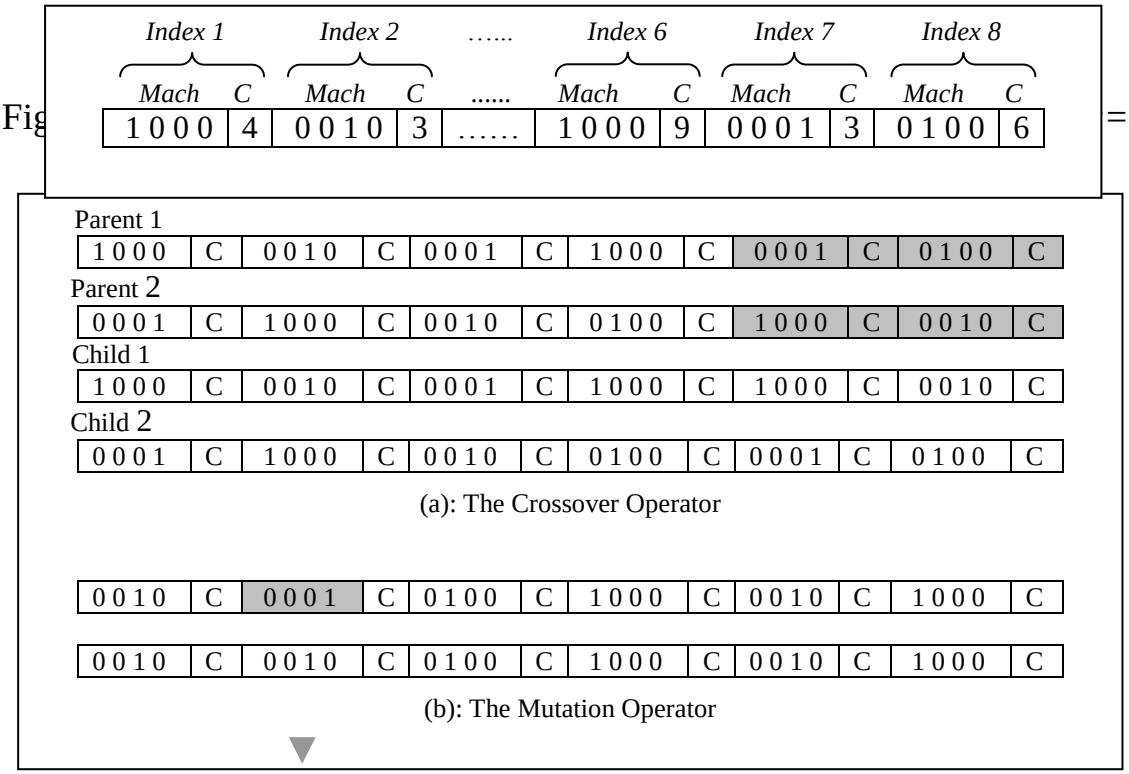

Figure 2: *Genetic operators for the problem*

Tables

Index	*Job No. (i)*	*Operation No. (k)*
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
6	2	3

Table 1*: Example of Encoding of Operation indices as Index*

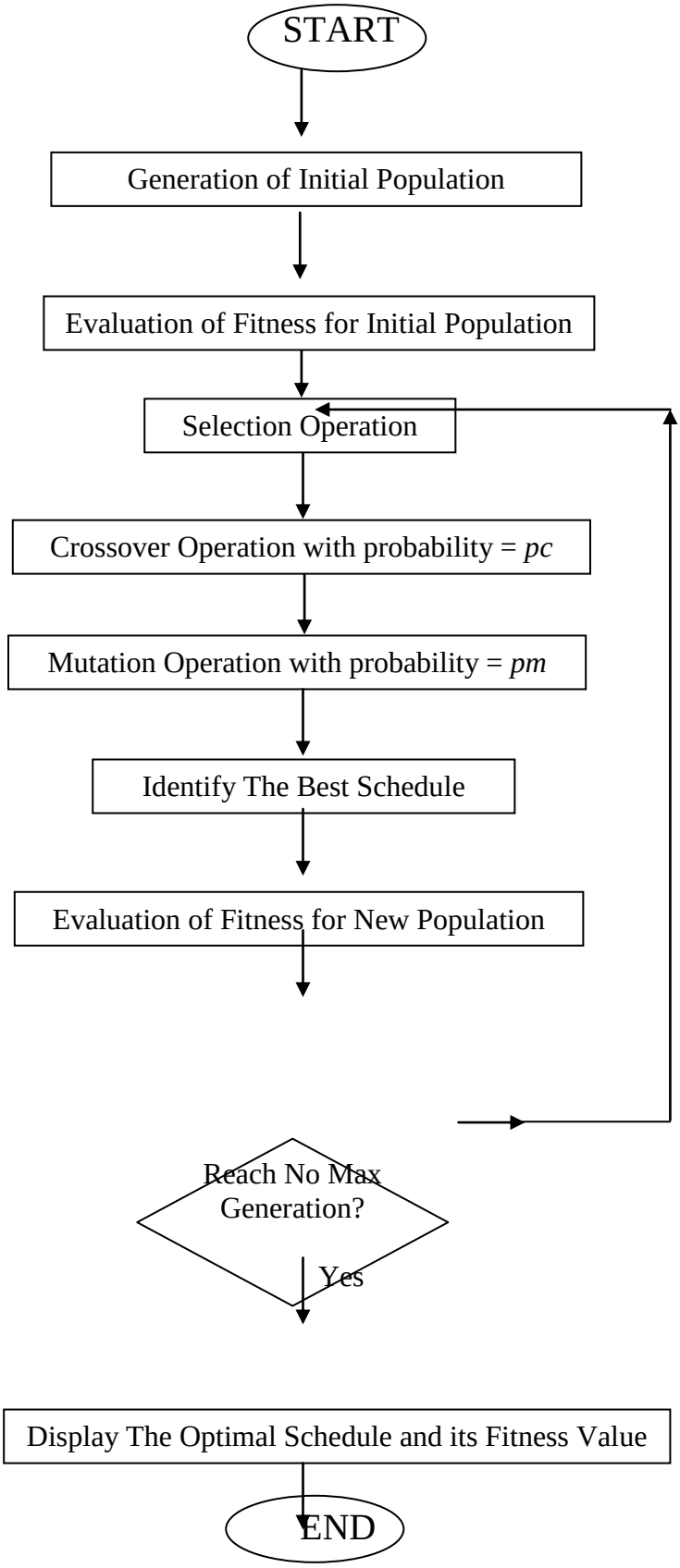

Figure 3: A Job Shop Scheduling Problem using Genetic Algorithm

	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6
Pc	0.7	0.7	0.8	0.85	0.7	0.85
Pm	0.001	0.001	0.001	0.001	0.001	0.01
Pop_Size	20	20	20	10	20	20
Max_Gen	50	80	50	100	50	50
No. of Machines	6	10	8	10	10	10
No. of Jobs	6	10	8	10	4	5

Table 2*: The Problem Parameters*

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	2,11	3,11	1,14	1,5	5,2	5,5
Job 2	5,8	5,1	4,13	1,5	2,12	6,11
Job 3	4,10	4,15	5,10	5,5	3,9	6,1
Job 4	3,2	3,3	2,1	6,5	5,6	6,10
Job 5	3,10	4,6	6,7	3,13	2,15	6,1
Job 6	1,13	5,1	1,8	6,2	2,11	4,3
The Makespan is 50						

Table 3: *Problem Instance 1*

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	2,13	4,5	10,3	1,7	7,1	2,1	7,15	8,3	7,11	7,5
Job 2	4,4	5,11	9,7	9,13	4,6	3,3	10,11	1,11	4,4	10,11
Job 3	1,1	3,3	1,9	9,8	6,2	9,5	7,4	1,12	9,14	8,5
Job 4	2,5	6,11	4,1	7,7	7,10	7,3	3,5	2,4	4,13	4,3
Job 5	6,12	3,9	8,15	7,2	9,13	2,6	8,13	8,10	6,9	9,8
Job 6	4,15	4,7	6,6	10,11	3,10	2,11	8,14	5,2	4,8	10,3
Job 7	4,4	3,9	3,9	7,7	8,4	5,12	6,13	8,2	9,1	4,2
Job 8	8,14	2,13	2,3	1,1	5,4	2,1	5,10	3,11	3,1	3,3
Job 9	10,5	1,12	5,5	2,5	4,5	2,8	8,2	1,11	10,7	3,14
Job 10	5,12	10,10	6,11	1,12	6,5	5,15	4,4	9,15	2,11	5,4
The makespan is 84										

Table 4: *Problem Instance 2*

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	6,2	1,10	5,4	5,14	5,3	8,10	7,5	1,1
Job 2	7,4	2,2	7,4	4,11	5,3	7,2	2,1	2,15
Job 3	3,3	8,10	5,1	3,3	6,8	3,6	8,12	4,9
Job 4	4,11	2,7	5,2	4,10	3,10	2,9	2,12	6,4
Job 5	7,3	5,1	3,13	3,4	1,12	1,10	3,12	7,7
Job 6	1,12	4,5	4,5	8,6	6,5	5,13	4,4	3,2
Job 7	3,3	7,11	5,10	7,3	6,5	5,8	6,1	5,2
Job 8	7,15	6,14	7,8	8,15	1,1	1,2	2,14	6,15
The makespan is 84								

Table 5: *Problem Instance 3*

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	8,8	1,15	7,12	8,3	6,6	8,4	2,7	7,2	10,12	4,13
Job 2	4,6	8,3	2,13	3,12	5,12	7,11	1,10	3,13	6,4	9,9
Job 3	5,10	2,3	10,8	10,5	5,15	10,11	2,10	6,10	4,1	2,6
Job 4	2,4	3,8	3,12	7,12	10,3	10,3	8,9	2,2	7,4	4,2
Job 5	8,13	2,7	1,5	7,6	9,11	2,9	6,10	5,1	8,1	4,9
Job 6	2,9	6,14	9,6	1,10	9,4	8,10	3,12	9,4	3,8	3,15
Job 7	2,3	9,9	4,11	10,3	10,11	1,15	4,11	6,1	3,6	10,6
Job 8	6,5	5,7	7,2	2,10	4,11	8,4	10,15	4,9	7,5	4,2
Job 9	7,7	4,5	3,2	9,14	8,6	7,6	5,9	2,5	5,11	5,7
Job 10	4,8	7,3	6,14	8,4	6,12	7,2	6,2	9,13	8,9	6,5
The Makespan is 88										

Table 6: *Problem Instance 4*

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	4,6	2,14	2,8	4,11	1,5	1,6	2,13	1,6	3,3	4,8
Job 2	1,8	4,8	1,14	2,15	1,10	1,8	1,4	4,11	1,1	4,14
Job 3	1,15	2,4	3,1	2,1	3,9	1,3	3,9	2,15	3,1	4,13
Job 4	3,10	2,1	4,10	3,3	4,12	1,7	2,7	2,4	3,3	3,10
Job 5	2,11	2,3	3,12	3,3	1,11	2,7	2,6	4,5	2,9	4,4
Job 6	4,11	3,1	3,15	1,4	3,2	4,10	2,1	1,9	3,9	1,1
Job 7	2,12	1,15	1,5	1,12	1,6	1,3	3,3	2,15	3,3	2,11
Job 8	4,15	4,13	3,10	2,13	1,1	2,14	4,8	4,14	4,2	1,6
Job 9	3,7	3,6	3,11	3,10	3,12	3,1	1,1	4,1	1,7	3,9
Job 10	1,5	1,4	4,5	3,14	4,13	3,2	1,10	3,12	1,1	2,10
The Makespan is 194										

Table 7: *Problem Instance 5*

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	2,10	4,12	1,9	4,5	3,6	2,11	4,7	1,12	1,4	2,4
Job 2	3,11	2,13	3,13	2,12	3,12	2,14	5,13	1,14	2,9	5,13
Job 3	4,6	5,13	4,10	5,9	1,11	3,10	4,5	1,9	5,13	4,5
Job 4	5,6	3,2	5,2	5,14	3,11	3,3	3,2	3,10	4,6	5,2
Job 5	1,14	2,9	4,12	5,3	3,3	1,10	1,5	5,11	3,2	3,5
Job 6	3,2	3,1	4,3	1,5	4,15	3,5	5,4	2,14	4,11	5,10
Job 7	4,2	2,1	1,8	4,12	5,14	4,1	4,1	5,9	2,6	4,5
Job 8	5,2	5,4	5,6	2,10	2,1	2,2	1,2	5,2	3,5	3,7
Job 9	2,14	3,5	4,3	4,1	1,12	1,6	4,10	1,4	5,5	3,15
Job 10	2,1	2,15	3,13	4,11	3,3	1,4	3,8	1,6	1,3	3,1
The Makespan is 155										

Table 8: *Problem Instance 6*

حل مشكلة جدولة الأعمال باستعمال الخوارزمية الجينية

واثق نجاح عبدالله
جامعة بغداد / كلية التربية – إبن الهيثم

الخلاصة

الجدولة الفعّالة للأعمال في القطاع الصناعي تكون مفيدة لتلبِية مطلبِ الإنتاج وتقليل كلفته، ولتَحسين القدرةِ للتَنَافُس في السوق المتقلبة المستمرة التزايد التي تَطلب منتجات متعدّدةَ.

في العديد من مشاكلِ تحقيقِ الأمثلية، مشاكل جدولة الأعمال إكتسبت سمعة لأن تكون صعبة الحل. جدولة الأعمال هي أساساً مشكلة ترتيب. النظام يقدم مخطط تشفير جديد لمشكلة جدولة الأعمال الكلاسيكية. إنّ الهدف هو ايجاد تخصيص لكل عمل وتحديد تتابع للأعمال على كل ماكنة بحيث يكون وقت إكمال الأعمال أقل ما يمكن.

الخوارزمية الجينية التي حققت نجاحاً كبيراً في إعطاء الحلول الكفوءة للعديد من مشاكل تحقيق الأمثلية غير المتعددة الحدود الصعبة أستعملت لحل مشكلة جدولة الأعمال. جداول الأعمال التي ولدتها الخوارزميات الجينية بُنيَت باستعمال قاعدة أولوية وتحت عدّة قيود. بعد الحصول على جدول الأعمال تجرى عملية تدقيق لضمان ان الحل عملي ومجدي. تم تجربة النظام على مجموعة من الحالات وأثبتت النتائج فعالية الخوارزمية.