Solving an assembly sequence optimisation problem using the genetic algorithm

1st Fawaz Alharbi
School of Engineering
University of Portsmouth
Portsmouth, United Kingdom
Fawaz.alharbi@port.ac.uk

2nd Qian Wang
School of Engineering
University of Portsmouth
Portsmouth, United Kingdom
Qian.wang@port.ac.uk

Abstract **If a product is made of more than one component, then it needs to be assembled. Complexity of assembling a product is largely subject to the implication of product design as well as the number of assembly components that may also affect attainable assembly sequences. In addition, it is widely understood that efficiency of assembling a product can be partially achieved by reducing assembly times (therefore costs) and this is notably important for small and medium-sized manufacturing companies. Thus, it is useful for seeking an optimal assembly sequence of a product at the early product design stage. This paper presents a study by applying the genetic algorithm (GA) for solving an assembly sequence optimisation problem of a car engine pump valve providing a quick solution in obtaining an optimal or near-optimal assembly sequence of the product.**

Keywords Assembly Sequence, Optimisation, Genetic Algorithm, Product Design

I. INTRODUCTION

An assembly sequence needs to be predefined by product designers aimed at reduction of assembly time therefore production costs, as it is crucial particularly for small-medium enterprises (SME) to survive in the fierce competitions of the global market. Chang et al stated that one of the problems in assembly sequence planning (ASP) is that the increasing number of components often leads to more constraints, which in turn make the assembly process more complex [1]. Ou et al adopted a matrix approach for analysing the information derived from a CAD model to obtain the assembly sequence for a two-stroke engine aiming to reduce both assembly time and cost [2]. Xing et al proposed a hybrid particle swarm genetic algorithm (PSGA) to generate the optimised assembly sequence [3]. Hongbo et al developed a genetic simulated annealing algorithm (GSAA) for solving an ASP optimisation problem [4]. Zhou et al presented the imperialist competitive algorithm used for seeking an optimal or near-optimal solution of an ASP [5]. Marian et al suggested a GA for solving an ASP optimisation problem with an aid of a guided search effective algorithm [6]. Choi et al developed an approach to optimise multi-criteria ASP based on a GA [7]. Yasin et al investigated the application of GA in optimising product assembly sequences and the study concluded that GA can be used to obtain a near optimal solution for seeking a minimal process time of sequence assembly [8]. Hong and Cho applied a GA to generate the optimal solution for a robotic assembly sequence aiming to minimise the assembly cost [9].

GA is an optimisation method based on the principle of the reproduction process of organism and it has been proved an effective approach used for solving optimisation problems of ASP due to its ability to offer a flexible way of defining constraints [10]. Marian described GA as an easy way of deriving from the fitness function as a penalty approach used to compute the fitness value for each possible assembly sequence. GA is often developed as a computer-based programming, which facilitates the optimisation of the proposed solutions [11]. GA starts with a generation of an initial population of called chromosomes that is a string of genes or symbols that can be coded; each chromosome represents a solution of the problem. By executing a number of pre-defined selection rules, the initial population evolves to be a population towards a final optimal solution selected after a series of successive iterations [5]. This paper presents a case study that applies the GA approach to obtain the fastest solution for the assembly sequence of a car engine pump valve.

II. THE GA MODEL FOR THE ASP OPTIMISATION

Development of a GA used for assembly sequence optimisation generally involves three steps: representation, generation, and optimisation. Representation can be categorised as two types: implicit and explicit. Implicit representation refers to precedence between two mating assembly parts. Explicit representation is involved in encoding possible assembly sequences with constraints. In this study, a population of possible assembly sequences was initially generated in a random manner. Such a generation refers to a creation of assembly sequences allowing a little perturbation during a crossover stage. Within a generation, the GA was able to select a subset of chromosomes (often two) from the current population, called parents for mating to create a new chromosome called a child or offspring. Optimisation is carried out by executing user-defined criteria to seek an optimal solution among generated assembly sequences. Termination of the GA optimisation process occurs after the entire search space is completed. The solution space was classified into families whereby a single family represented a single valid assembly sequence [12]. A chromosome that contains a parent assembly sequence is probabilistically selected based on an evaluation of fitness relating to the current population. Only the chromosome (containing a solution) with a higher fitness value has a greater chance to be selected for mating with another chromosome with a higher fitness value to produce a new chromosome. The crossover is a process that carries out an exchange of parental genes for creating a new chromosome. Genetic operators are subsequently applied leading to a generation of an offspring sequence. A transformation function replaces an invalid offspring with a family of valid assembly sequences. Further, genetic diversity can be introduced into the chromosomes of a population or family using crossover and mutation to generate a family of new chromosomes and the GA repeatedly compares the fitness value of a chromosome with another until the optimal chromosome is formed [13]. Fig. 1 illustrates the GA programming approach used in this study.

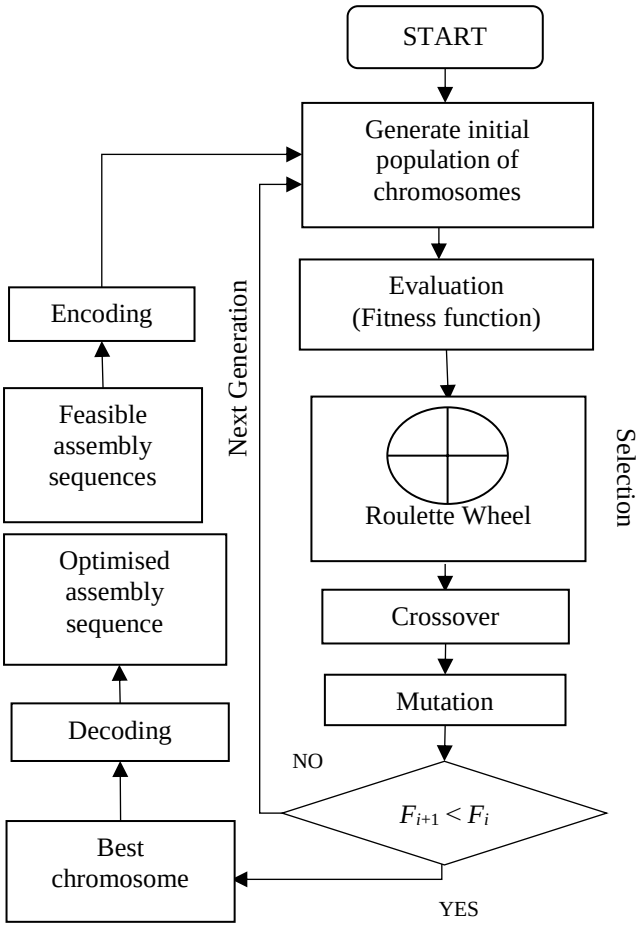

Fig. 1. The GA programming approach

As shown in Fig.1, the initial population is usually generated randomly as a binary string of zeros and ones or as integers; this is also known as a genetic representation or encoding. The next process is the evaluation stage, which involves a computation of a fitness value based on an objective (fitness) function. Thus, selection plays a key role in GA programming; only those representing a possible solution with either a highest or lowest fitness value that are selected. The Roulette Wheel approach was used to ensure that a certain number of the population of chromosomes are retained in the next generation, which contains chromosomes with greater fitness. After the selection, crossover is usually carried out via a random selection of parental chromosomes to produce new chromosomes. Crossover normally operates on pairs of chromosomes simultaneously with the aim of creating offspring that combines the features of both parental chromosomes. In this study, however, it was performed by crossing over the genes as illustrated in Fig.3 to generate possible assembly sequences for the car engine pump valve; assuming that the bits of chromosomes can be swapped freely without following the precedence required for assembly. The last process is the mutation, which is used to have a complete loss of a particular allele or bit, i.e, the mutation of swapped genes is utilized to prevent chromosomes from repeating the gene of a new offspring. In this study, this was performed by crossing over the genes in different sequences leading to various assembly paths and total time of assembly. Only the bits of chromosomes that do not have a successor or precedence are swapped as illustrated in Fig.3 and these chromosomes were used.

III. CASE STUDY

Table 1 shows a list of components used for assembly of a car engine pump valve used as a case study of this work. Fig. 2 shows the drawing of assembly parts of the car engine pump valve. Table 2 shows the liaisons between two possible assembly components of the car engine pump valve. Table 3 shows the average time taken for assembly between two possible components.

TABLE I. ASSEMBLY COMPONENTS OF THE CAR ENGINE PUMP VALVE

Component Number	Component Names
1	Arm
2	Body
3	Bolt
4	Bolt-Shaft
5	Key
6	Nut-Shaft
7	Nut3
8	Plate
9	Retainer
10	Shaft
11	Sleeve1
12	Sleeve2
13	Washer-shaft
14	Washer3

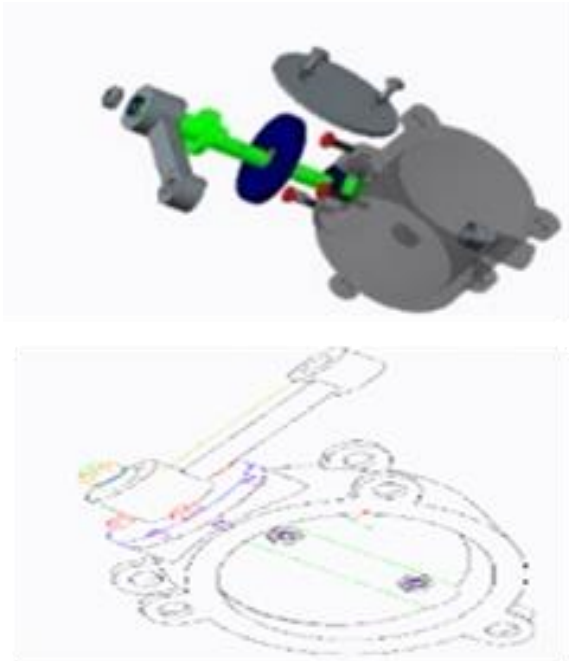

Fig. 2. Components of the car engine pump valve

TABLE II. THE PRIORITY MATRIX IN LIAISONS BETWEEN TWO POSSIBLE ASSEMBLY COMPONENTS OF THE CAR ENGINE PUMP VALVE

Comp.	***Sleeve (11)***	***Plate (8)***	***Nut-Shaft (6)***	***Bolt-Shaft (4)***	***Washer-Shaft (13)***	***Shaft (10)***	***Body (2)***	***Arm (1)***	***Sleeve (12)***	***Retainer (9)***	***Bolt (3)***	***Key (5)***	***Washer (14)***	***Nut (7)***
Sleeve (11)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Plate (8)	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Nut-Shaft (6)	1	1	0	0	0	1	0	0	0	0	0	0	0	0
Bolt-Shaft (4)	1	1	0	0	0	1	0	0	0	0	0	0	0	0
Washer-Shaft (13)	1	1	0	0	0	1	0	0	0	0	0	0	0	0
Shaft (10)	1	1	0	1	0	0	0	0	0	0	0	0	0	0
Body (2)	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Arm (1)	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Sleeve (12)	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Retainer (9)	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Bolt (3)	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Key (5)	1	1	1	1	1	1	1	1	1	1	1	0	0	0
Washer (14)	1	1	1	1	1	1	1	1	1	1	1	0	0	0
Nut (7)	1	1	1	1	1	1	1	1	1	1	1	0	0	0

TABLE III. AVERAGE ASSEMBLY TIME BETWEEN TWO POSSIBLE COMPONENTS OF THE CAR ENGINE PUMP VALVE

Comp.	***Sleeve (11)***	***Plate (8)***	***Nut-Shaft (6)***	***Bolt-Shaft (4)***	***Washer-Shaft (13)***	***Shaft (10)***	***Body (2)***	***Arm (1)***	***Sleeve (12)***	***Retainer (9)***	***Bolt (3)***	***Key (5)***	***Washer (14)***	***Nut (7)***
Sleeve1 (11)	0	2	2	1	1	3	4	2	3	1	4	5	5	4
Plate (8)	2	0	5	2	2	6	6	3	10	3	2	2	5	2
Nut-Shaft (6)	3	3	0	2	2	3	3	1	3	4	5	3	4	5
Bolt-Shaft (4)	2	5	5	0	11	15	4	4	4	4	4	5	8	2
Washer-Shaft (13)	4	4	10	10	0	7	13	2	5	6	5	4	6	3
Shaft (10)	3	5	2	7	7	0	2	13	7	8	6	6	4	5
Body (2)	4	8	1	3	3	4	0	3	18	7	7	7	6	8
Arm (1)	6	7	2	8	8	5	6	0	6	6	4	3	5	6
Sleeve (12)	8	6	4	5	5	8	7	17	0	52	2	4	7	4
Retainer (9)	9	8	6	2	2	7	18	7	4	0	42	2	8	7
Bolt (3)	7	6	8	8	8	13	6	5	3	3	0	5	5	5
Key (5)	4	14	18	7	7	4	4	3	6	4	5	0	4	4
Washer (14)	2	6	6	2	9	6	2	4	1	5	6	4	0	1
Nut (7)	4	3	5	4	4	5	3	4	8	7	3	4	1	0

Note: Assembly time is calculated in seconds.

The following notations and parameters are used:

i : Number of a chromosome, i = 1, 2, 3,…, k
fi: Fitness of chromosome i
ti : Time taken of chromosome i
F : Fitness of the population
Cr : Crossover rate
Ri : Roulette Wheel probability
Pi : Cumulative probability for chromosome i
L : Length of a gene
e : A gene in a chromosome, n = 1, 2, 3, …, n
mr : Mutation rate
r : Random number
M : Number of mutations

A. The Fitness Function

In this study, the GA uses the single objective function as the fitness function for selecting a chromosome with a higher fitness value. The fitness *fi*, which is a function of assembly time of an assembly sequence represented by chromosome *i*, is described as:

$$f_i = \frac{1}{\sum_{i=1}^{n} ti} \quad (1)$$

Thus, the total fitness *F* is given:

$$F = \sum_{i=1}^{n} f_i \quad (2)$$

B. *Selection of a chromosome*

The fitness proportionate selection, which is also known as the roulette wheel selection, fitness is calculated by assigning a fitness value to one of possible chromosomes or solutions. This fitness value is often associated with a probability of a selection with each of individual chromosomes. Only a chromosome with a high fitness value will be selected during a selection process. Thus, only a chromosome, which is fittest with the greater roulette wheel probability, is selected. The roulette wheel probability *Ri* is given by:

$$R_i = \frac{f_i}{F} \tag{3}$$

The percentage of the chance for chromosome *i* is expressed as probability P_i where,

$$P_i = R_i \tag{4}$$

C. *Crossover*

Fig.3. illustrates a crossover process where the first two genes of two different chromosomes are exchanged. The crossover process is controlled by a probabilistic operator. Repetition of the same gene number is strictly avoided during the crossover process and each of the genes involved is thoroughly checked before completing the process.

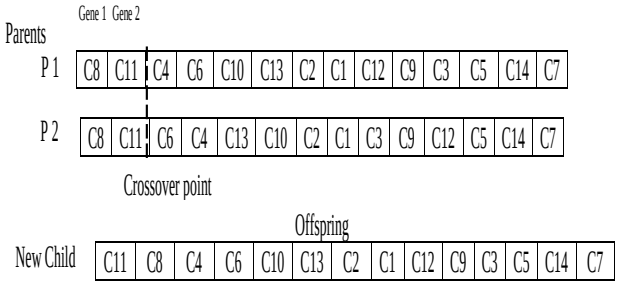

Fig. 3. The crossover process of swapping genes

D. *Mutation*

Mutation is performed by a random replacement of a gene from its original state with a new quantity in other position or attribute according to a user-defined mutation probability or mutation rate. This avoids taking the fittest only of a population in generating the next but a random selection with a weighting towards those that are fitter. In this study as an example, parameters of C8 and C11 were used for the calculation of the mutated chromosomes in a particular population as shown in Fig.3. Thus, the length *L* of gene n in a chromosome *i* is thereby given by:

$$L = e_n^i \tag{5}$$

As a result of this, a probability of a mutation of a gene is 1/*L*, If the mutation rate *mr* is greater than the selected random number r, i.e., *mr* > *r*, where is r a random number *r* in the range between 0 and 1, then the mutation should be performed. Hence, the number of mutations *M* is given by:

$$M = rL \tag{6}$$

After a re-allocation of the suitable gene position of the chosen parent, a new child chromosome is established. This also implies that the new child chromosome has a new identification which possibly makes it a new parent for the next generation of continuous population.

IV. RESULTS

Fig.4. shows the results obtained from 5 generations using the GA approach. Each result shows the assembly time in response to each of 10 chromosomes, of which each depicts a possible assembly sequence for the car engine pump valve. As shown in Fig.4, generation 1 has the highest assembly time, which is 570 seconds. The assembly time obtained in generation 2 decreases to be 560 seconds. The results obtained in generation 3, 4 and 5 respectively is the lowest assembly time, which is 500 seconds.

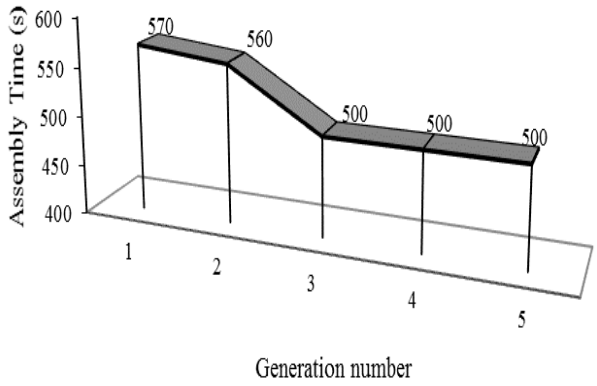

Fig. 4. Assembly time obtained in response to generation number using the GA

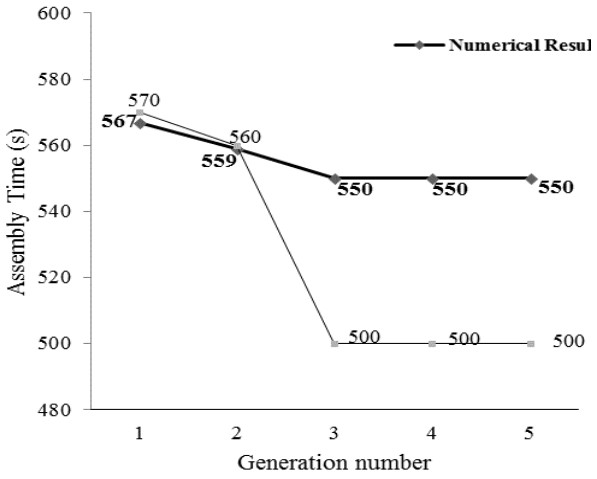

Fig. 5. Comparison in assembly time between the numerical result and computerised result using the GA

V. CONCLUSIONS

This paper presented an investigation into feasibility and applicability using the GA approach for solving the assembly sequence problem of a car engine pump valve. The paper illustrates the mechanism of the GA programming model used for determining an assembly sequence with the minimal assembly time. The aim of this study was to reduce assembly time by developing the GA algorithm implemented in Java. The results show that this is a useful decision-making tool in

obtaining an optimal or near-optimal assembly sequence of for product designers providing a quick solution for seeking an optimal assembly sequence in a cost-effective way. Nevertheless, in practice, assembly work is often performed by human workers and therefore the assembly system has a random behaviour that also impact on assembly time. Thus, the computer modelling simulation approach by implementing the GA may need to be applied.

REFERENCES

[1] C.C. Chang, H.E. Tseng, L.P. Meng, Artificial immune systems for assembly sequence planning exploration, Eng. App. of Artif. Intell, 22, pp. 1218-1232, 2009.

[2] L.M. Ou, X. Xu, Relationship matrix based automatic assembly sequence generation from a CAD model, Comp.-Aided Design, 45, pp. 1053-1067, 2013.

[3] Y. Xing, Y. Wang, Assembly sequence planning based on a hybrid particle swarm optimisation and genetic algorithm, Int. J. of Prod. Res, 24, pp. 7303-7312, 2012.

[4] S. Hongbo, L. Shuxia, G. Degang, L. Peng, Genetic Simulated Annealing Algorithm-Based Assembly Sequence Planning, International Technology and Innovation Conference, pp. 1573-1579, 2006.

[5] W. Zhou, J. Yan, Y. Li, C. Xia, J. Zheng, Imperialist competitive algorithm for assembly sequence planning, Int. J. of Adv. Manuf Tech. 67, pp. 2207–2216, 2013.

[6] R. M. Marian, L.H.S. Luong, K. Abhary, A genetic algorithm for the optimisation of assembly sequences, Computers & Industrial Engineering, 50, pp. 503–527, 2006.

[7] Y.K. Choi, D.M. Lee, Y.B. Cho, An approach to multi-criteria assembly sequence planning using genetic algorithms, Int. J. of Adv. Manuf. Tech. 42, pp. 180-188, 2009.

[8] A. Yasin, N. Puteh, R. Daud, M. Omar, S.L.S. Abdullah, Product assembly sequence optimisation based on genetic algorithm, Int. J. on Comp. Sci. and Eng. 29, pp. 3065-3070, 2010.

[9] D.S. Hong and H.S. Cho, A genetic-algorithm based approach to the generation of robotic assembly sequence, Control Engineering Practice, 7, pp. 151-159, 1999.

[10] D. Whitley, An executable model of a simple genetic algorithm. Foundations of genetic algorithms, 2 (15-19), pp. 45-62, 2014.

[11] R.M. Marian, L.H.S. Luong, K. Abhary, Assembly sequence planning and optimisation using genetic algorithms Part I. Automatic generation of feasible assembly sequences, Applied Soft Computing, 2/3F, pp. 223–253, 2003.

[12] N. Senin, R. Groppetti, D.R. Wallace, Concurrent assembly planning with genetic algorithms, Robotics and Computer Integrated Manufacturing, 16, pp. 65-72, 2000.

[13] Alharbi, F. S., & Wang, Q., A genetic algorithm for solving assembly sequence problem, Applied Mechanics and Materials, Vol. 872, pp. 420-424, 2017.