A genetic algorithm to optimize the manufacturing process of a robotic arm under fuzziness

P. T. Zacharia · S. A. Tsirkas · G. Kabouridis · A. Ch. Yiannopoulos · G. I. Giannopoulos*

Department of Mechanical Engineering, Technological Educational Institute of Western Greece, Megalou Alexandrou 1, 26334 Patras, Greece

This paper presents an application of the fuzzy assembly line balancing problem of type 2 (SALBP-2), which is an NP-hard problem. The optimization tool for the solution of the fuzzy SALBP-2 is a Genetic Algorithm (GA). The efficiency of the proposed approach is tested on the construction process of a robotic arm. Bearing in mind that the data obtained from more realistic situations are imprecise and uncertain, the consideration of fuzziness for the solution of SALBP-2 is of great interest. Thus, real data values for the processing times are gathered and estimated as within uncertainty. Then, fuzzy processing times are used for finding the optimum cycle time needed for finishing the construction of the robotic arm. The experimental results demonstrate the effectiveness and efficiency of the proposed GA in determining the optimum sequence of the tasks assigned to workstations and providing the optimum fuzzy cycle time.

Keywords: fuzzy numbers; genetic algorithm; production; process planning; machine tools; metal parts

---

*Corresponding author. E-mail: ggiannopoulos@teiwest.gr

1. Introduction

In mass production systems, an important problem is the assembly line problem. An assembly line is a manufacturing technique according which parts are added in sequence from workstation to workstation until the final assembly is produced. Each station has to complete a set of tasks on parts moving along the line. There are numerous studies related with assembly line systems which focus on the determination of the set of tasks which have to be assigned to each workstation under a given cycle and the constraints of precedence relationships. This kind of problem is well known as the simple assembly line balancing problem (SALBP) (Scholl and Becker 2006).

The most famous versions of the abovementioned problem is the SALBP type 1 (SALBP-1) and the SALBP type 2 (SALBP-2). SALBP-1 (Goncalves and Almeida 2002; Lapierre, Ruiz, and Soriano 2006; Kilincci and Bayhan 2006; Zhang et al. 2007; Nearchou 2008; Kilincci and Bayhan 2008; Yeh and Kao 2009; Sulaiman, Choo, and Chong 2011; Dou, Li, and Lv 2011; Dou, Su, and Li 2011; Fathi et al. 2011; Ariffin, Fathi, and Ismail 2012; Dou, Li, and Su 2013; Atasagun and Kara 2014) is present when the aim is to effectively assign tasks to workstations by minimizing the number of stations for a pre-specified cycle time. This problem commonly arises when new assembly lines are to be designed by a company. On the other hand SALBP-2 (Zhang et al. 2007; Gu et al. 2007; Kilincci 2010; Blum 2011; Tang et al. 2011; Avikal, Mishra, and Jain 2012; Zheng et al. 2013) is present when the aim is to minimize the cycle time for a specific number of stations. This kind of problem usually arises when changes in the production process of a product are to take place in the effort to improve the line efficiency. Several methods have been proposed for the solution of SALBP problems such as genetic algorithms (GAs)

(Goncalves and Almeida 2002; Zhang et al. 2007; Gu et al. 2007), ant colony optimization (Zhang et al. 2007; Sulaiman, Choo, and Chong 2011; Zheng et al. 2013), particle swarm optimization (Dou, Li, and Lv 2011; Dou, Su, and Li 2011; Dou, Li, and Su 2013), Petri net (Kilincci and Bayhan 2006; Kilincci and Bayhan 2008; Kilincci 2010), tabu search (Lapierre, Ruiz, and Soriano 2006), bacterial foraging optimization (Atasagun and Kara 2014) or other heuristic algorithms (Fathi et al. 2011; Ariffin, Fathi, and Ismail 2012; Yeh and Kao 2009; Blum 2011; Tang et al. 2011; Avikal, Mishra, and Jain 2012).

In a realistic manufacturing environment, the task time maybe random due to worker fatigue, low skill levels job dissatisfaction, poorly maintained equipment, defects in raw materials etc. Since data in real-world problems are often afflicted with uncertainty, imprecision and vagueness due to both machine and human factors, they can only be estimated as within uncertainty. Several researchers have been attempted to incorporate fuzzy information in their effort to solve SALBP through various types of algorithms. Kalender ,Yilmaz, and Turkbey (2008) have developed an algorithm to solve traditional assembly line balancing problem (ALPB) with fuzzy operation times. Ozcan and Toklu (2009) have presented a fuzzy goal programming model for imprecise goals for two-sided assembly line balancing. Tapkan, Ozbakir, and Baykasoglu (2012) have solved two-sided ALPB by employing positional, zoning and synchronous task constraints via a bees algorithm. Mutlu and Ozgormus (2012) have considered the physical workload of a task as a fuzzy concept and proposed a fuzzy linear programming model to solve ALPB. La Scalia et al. (2013) have used of fuzzy set theory as a viable alternative method for modelling and solving the stochastic ALPB. In several attempts, GAs have been adopted to solve SALBP in conjunction with fuzzy logic. Tsujimura, Gen, and Kubota (1995) have illustrated via a numerical

experiment that a genetic algorithm (GA) is an appropriate tool to solve fuzzy scheduling problems. In an another attempt, Gen, Tsujimura, and Li (1996) have used a numerical example to solve ALPB with fuzzy processing time by using GAs with the objective of minimizing the total operation time in each work station. Similarly, Khoshalhan and Zegordi (2003) have presented a GA based approach for both types of fuzzy ALPB. Zacharia and Nearchou (2012) have presented a fuzzy extension of the SALBP-2 with fuzzy job processing times to deal with the uncertainty occurred in production systems. Rajabalipour Cheshmehgaz et al. (2012) have modeled ALBP through a multi-criteria fuzzy-GA.

In the present paper, the SALBP-2 regarding the construction of a real robotic arm is under investigation. The metal parts of the robotic arm are manufactured in four machining workstations. In order to deal with the variability in task operation times, fuzzy set theory (Zadeh 1965) is adopted as a very promising approach for modeling and solving stochastic problems. The fuzzy theory is then combined with an appropriate GA (Holland 1975; Goldberg 1989) for solving the fuzzy SALBP-2 for the assembly line of the robot's metal frame. To handle more realistically the manufacturing of the robot's metal frame, the processing time for each job is considered as fuzzy and is represented by triangular fuzzy membership functions. In an attempt to treat relevant imprecise data, fuzzy numbers are introduced to represent the processing time of each job, where the membership function of a fuzzy data represents the grade of satisfaction of a decision maker. The main contribution of this paper is that an effort is being made to enhance the real manufacturing process of a robotic arm in terms of time reduction using an optimization algorithm (GA) by simultaneously considering the variability and ambiguity associated with real situations.

The rest of the paper is organized as follows: Section 2 summarizes the arithmetics of fuzzy numbers, gives the background of the fuzzy assembly line balancing of type 2, and presents the practical problem of manufacturing the robot components. Section 3 presents the proposed optimization approach applied for the fuzzy assembly line balancing of the robotic arm. Computational results concerning the performance of the GA are presented in Section 4, while conclusions and directions for future work are pointed out and discussed in Section 5.

2. The fuzzy assembly line balancing problem

2.1. Arithmetic and ranking fuzzy numbers

Compared to traditional binary sets (where variables may take on true or false values), fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false.

A fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership (characteristic) function which assigns to each object a grade of membership ranging between zero and one. The membership function which represents a fuzzy set $\tilde{A}$ is usually denoted by μ_A. For an element x, the value $\mu_A(A)$ is called the membership degree of x in the fuzzy set $\tilde{A}$. The membership degree $\mu_A(A)$ quantifies the grade of membership of the element x to the fuzzy set $\tilde{A}$. The value 0 means that x is not a member of the fuzzy set; the value

1 means that x is fully a member of the fuzzy set. The values between 0 and 1 characterize fuzzy members, which belong to the fuzzy set only partially.

Due to the nature of processing times, the most commonly used fuzzy sets in depicting these values are triangular fuzzy numbers (TFNs) (Fonseca et al. 2005), that establish extreme points to represent the most likely and least likely values for the individual variables. In this work, the time variables are now represented as TFNs. TFN $\tilde{A}$ is denoted as a triplet of points , i.e. $\tilde{A}=(\alpha_1,\alpha_2,\alpha_3)$, where $\alpha_1<\alpha_2<\alpha_3$. In the adapted fuzzy heuristics, the tasks' fuzzy processing times are accumulated using the fuzzy addition operator. In particular, by assuming a second TFN $\tilde{B}=(\beta_1,\beta_2,\beta_3)$ where $\beta_1<\beta_2<\beta_3$, then the following arithmetics between $\tilde{A}$ and $\tilde{B}$ may be performed (Kaufmann and Gupta 1985):

$$\begin{aligned}\tilde{A}+\tilde{B}&=(\alpha_1+\beta_1,\alpha_2+\beta_2,\alpha_3+\beta_3)\\ \tilde{A}-\tilde{B}&=(\alpha_1-\beta_3,\alpha_2-\beta_2,\alpha_3-\beta_1)\\ \tilde{A}\times\tilde{B}&=(\alpha_1\cdot\beta_1,\alpha_2\cdot\beta_2,\alpha_3\cdot\beta_3)\\ \tilde{A}/\tilde{B}&=(\alpha_1/\beta_3,\alpha_2/\beta_2,\alpha_3/\beta_1)\end{aligned} \quad (1)$$

To compare the fuzzy numbers, some criteria to rank the fuzzy sets. The ranking method in this work involves three ordered criteria (Kim, Kim, and Kim 1996) which are explained in the following.

The greatest associate ordinary number F_1 is used as a first criterion for ranking:

$$F_1(\tilde{A})=\frac{\alpha_1+2\alpha_2+\alpha_3}{4} \quad (2)$$

If F_1 does not rank the fuzzy numbers then those which have the best maximum presumption F_2 (the mode) will be chosen:

$$F_2\left(\tilde{A}\right)=\alpha_2 \tag{3}$$

If F_1 and F_2 do not rank the fuzzy numbers then the divergence F_3 (the distance between two end-points) will be used as third criterion:

$$F_3\left(\tilde{A}\right)=\alpha_3-\alpha_1 \tag{4}$$

Consider a set *Q* composed of the TFNs $\tilde{A}_i$, $i=1,2,....,n$. A TFN is called major and denoted as $\tilde{A}^*$ when dominates all the others in some criterion, in *Q*, that is, $\tilde{A}^*=\max Q$ (the operator max is the discrete maximum). The decision maker chooses some criteria and determines its order of dominance. If the first criterion can not determine the major TFN then the second criterion follows and so on. On the contrary, a TFN is called minor when dominated by all others in *Q* and this operation is represented as min.

2.2. Fundamentals of fuzzy assembly line balancing of type-2

The fuzzy SALBP can be stated as follows: m workstations are arranged along an assembly line. Manufacturing a single product on the line requires the partitioning of the total work into a set $V=\{1,...,n\}$ of n elementary operations called tasks. Each

task j is performed on exactly one station and requires a fuzzy processing time $\tilde{t}_j$. Let $S_z\,(z=1,...,m)$ be the station load of station z (i.e. the set of tasks assigned to z), with a cumulated fuzzy task time $\tilde{t}S_z = \sum_{j \in S_z} \tilde{t}_j \;\; (z=1,...,m)$. The tasks are partially ordered by precedence relations defining a directed acyclic graph (DAG) $G=(V,E)$ with V being the set of the nodes denoting the tasks in G while E being the set of the edges representing the precedence constraints among these tasks. The assembly line is associated with a fuzzy cycle time $\tilde{c}$ denoting the maximum processing time available for each station. The aim of SALBP-2 is the minimization of the fuzzy cycle time $\tilde{c}$ (i.e. the maximization of the production rate) given the number of workstations m.

3. The proposed optimization approach

GAs (Holland 1975; Goldberg 1989) are optimization techniques which simulate the natural selection mechanism observed in the biological evolution process. A GA has global and parallel search capability while is suitable for solving demanding problems of high nonlinearity. A GA, in contrast with common search techniques, starts with an initial set of random solutions called population (individuals) which satisfy the constraints of the problem. Chromosome is called each individual in the population representing a solution to the problem at hand. Each chromosome comprises a number of structures known as genes. The chromosomes are then regressed via iterations known as generations. During each iterative procedure, i.e. generation, the chromosomes are estimated by utilizing some measures of fitness. This means that in every generation, the fitness of every individual in the population is calculated. Then the more fit individuals are selected from the current population, and each individual's

genome is changed to form the population of the next generation. This new population is then used in the next iteration of the algorithm. The next generation is created according to the fitness values by forming new chromosomes. These chromosomes arise by merging two chromosomes from current generation using a crossover operator or by changing a chromosome utilizing a mutation operator. The complete set of chromosomes is called a genotype, and the resulting organism is called a phenotype. Generally, the GA after numerous generations converges to the best chromosome, which represents the ideal solution to the problem. A typical GA includes a genetic representation of the solution domain and an efficient fitness function to evaluate the solution domain.

The proposed GA for the requirements of the present study has the following basic components: (a) The representation mechanism which is a method to transform phenotypes into genotypes, (b) the decoding mechanism which is a method to map phenotypes to solutions, (c) the evaluation mechanism which is a method to compute the cost-function for each genotype, (d) the mechanism which generates the initial population of the genotypes, (e) the mechanism which generates new genotypes by applying operators on the entire population, (f) the control parameters and (g) the termination condition. The block diagram of the present GA is illustrated in Figure 1 while its main steps are explained with more detail in the following sub-sections.

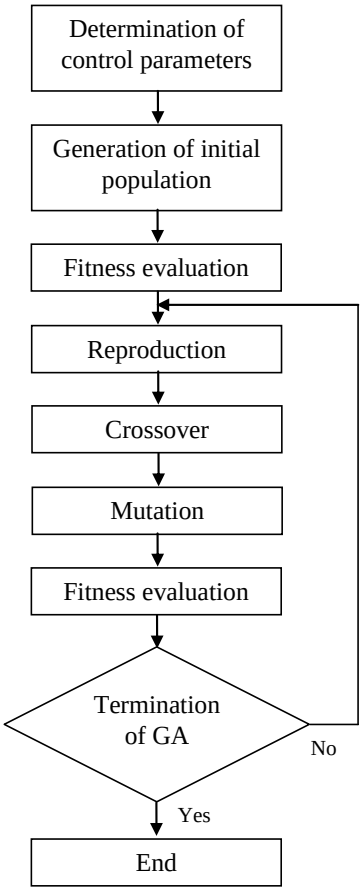

Figure 1. The block diagram of the proposed GA.

3.1. The representation mechanism

A GA can only find possible solutions to a problem when the solutions are transformed into a representation which the GA may handle. Thus, an appropriate representation mechanism is required to transform possible solutions within the context of the original problem, called phenotypes, into individuals within the context of the GA, called genotypes. This encoding may be realized via a string of binary code, real-valued numbers, integers, or a tree structure. Nevertheless, for mechanical engineering problems the most efficient representation is a string of real-valued numbers. Thus, in the present study a real-valued GA has been adopted in which genotypes are represented by floating-point vectors. Since ALBP solutions are represented by strings of integers (Scholl and Becker 2006) the utilized representation

mechanism should allow mapping from the genotypic state-level (the real-valued vectors) to the phenotypic level (the actual ALB solutions). This is realized by a simplified however efficient topological ordering scheme which is based on the relative priorities imposed by the components of a genotype. Assuming the n task of the ALBP with precedence relations given by a DAG $G=(V,E)$, the utilized representation mechanism aims to generate a topological sort of G from a specific n-dimensional floating-point vector ψ (genotype). Each vector's component ψ_i ($i=1,2,...,n$) represents the relative priority of task i ($i \in V$). The topological sort is therefore a ranking of all the tasks in line with their priorities and precedence constraints. During the procedure, the tasks with no predecessors are identified and put in set V'. Then, the task in V' which has the highest gene's value in ψ is removed from V' and placed in the next available position of a new string (initially empty) called partial schedule (PS). The process is continued until the completion of PS is achieved.

3.2. The decoding mechanism

After encoding a specific genotype into a phenotype, the decoding mechanism is required to assign the tasks in the generated task-sequence into the stations. The proposed technique by Kim et al. (Kim, Kim, and Kim 1996) seems to be more effective in contrast with other traditional schemes (Scholl and Becker 2006) and therefore preferred in the proposed GA. The utilized scheme consists of the following steps:

- Step 1: Set $\tilde{c}$ equal to the theoretical minimum fuzzy cycle time, i.e. $\tilde{c}_{th} = \tilde{t}_{sum}/m$.

- Step 2: Assign as many tasks as possible into the first *m-1* workstations. Assign all the remaining tasks to the last workstation, *m*.
- Step 3: Calculate the fuzzy work load $\tilde{W}_z$ for each workstation *z* (*z*=1, ...,*m*), and the potential fuzzy workload $\tilde{PW}_z$ (*z*=1,...,*m*-1), where $\tilde{PW}_z$ is the sum of $\tilde{t}S_z$ and the processing time of the first task assigned to (*z*+1)-st station (*z*=1,...,*m*-1).
- Step 4: Set $\tilde{c}_w = \max\left\{\tilde{W}_1, \tilde{W}_2, ..., \tilde{W}_m\right\}$ and $\tilde{c} = \min\left\{\tilde{PW}_1, \tilde{PW}_2, ..., \tilde{PW}_{m-1}\right\}$.
- Step 5: If $\tilde{c}_w > \tilde{c}$, then go to Step 2 else Return $\tilde{c}_w$

3.3. The evaluation mechanism

The evaluation mechanism corresponds to the computation of the objective function $\tilde{c}$ for each phenotype of the current population. The objective with SALBP-2 is to minimize the fuzzy cycle time $\tilde{c}$. The objective function has to be transformed to the fitness function f, which is evaluated for all the chromosomes of the population. The value of the fitness function for one chromosome expresses its ability to survive and be reproduced in the next generation. The fitness function is the inverse of the objective function:

$$f = \frac{1}{\tilde{c}} \tag{5}$$

Equation (5) aims to maximize the fitness function and consequently forcing $\tilde{c}$ to a minimum value.

3.4. The initial population

In order to initialize the process, the solutions in the first generation have to be defined. Commonly, this is done randomly since the main desire is to spread the first individuals over the complete search space before converging to more promising regions. Nevertheless, when the area of the optimum solution may be estimated beforehand then the algorithm could be initiated around this area to speed up the convergence.

3.5. The genetic operators

A genetic operator is used in genetic algorithms to maintain genetic diversity. Genetic operators such as crossover, mutation and selection are used in GAs to assure genetic variation for the process of evolution. Roulette wheel selection, is the best known selection operator and thus adopted here. The concept is to evaluate selection probability for each chromosome proportionally to the fitness value. Then a model roulette wheel is made which display these probabilities. The selection process is based on spinning the specific wheel the number of times equal to population size. Crossover is an operator which aims to produce new individuals by joining parts of several individuals of the previous generation. The selection of individuals is made randomly according to a predefined probability, i.e. one-point crossover rate (Michalewicz 1996). Mutation is required to insert new genetic material into the population by slightly modifying the genotype representation. In this way, the early convergence to local minima is avoided. The modification applied stochastically to discover potential better solutions based on the current best solutions. The mutation operator is applied by changing a random gene (i.e. a floating number) according to a small-predefined probability (mutation rate).

3.6. Control parameters

The most important control parameters are the population size, the crossover ratio, the mutation rate and the elitism.

Population size is one of the important topics to consider in evolutionary computation since small population size may lead the algorithm to poor solutions while a large population size may require a much more computational time to find a solution. The population size selection should be made according to the nature and the complexity of the current problem. In the present work, a variety of tests involving the influence of the population size have been made and finally the population size has been set to 100 for the SALBP-2 under investigation which involves 4 workstations and 57 tasks.

The crossover rate controls the frequency with which the crossover is applied. The higher crossover rate, the more quickly new individuals are added to the population. Several convergence tests have proved the effectiveness of a crossover rate equal to 0.8 for the purpose of the present research.

The mutation rate defines the probability according which the position of each individual in the intermediate population undergoes a random change. A GA with a too high mutation rate will inevitably become a random search. Thus, the mutation rate is typically chosen to be less than 0.4. The value of 0.1 has been chosen here.

When creating new population via crossover and mutation, there is a chance that the best chromosome will be lost. Elitism is a method according which the best chromosome is copied to the new population. A comparison involving the best chromosome between current and previous generation is required. Elitism increases GA performance since it prevents the loss of the best-found solution.

3.7. Termination conditions

The termination condition should be theoretically satisfied and thus end the algorithm when the optimum solution has been found. However, for many optimization problems the ideal solution is unknown and therefore there is always an uncertainty whether better solutions exist. In addition stochastic procedures may require a significant computational cost and take a long time to converge to the optimum solution. Commonly used termination conditions are therefore based on the maximum allowed number of iterations (generations) which however present limitations since it is difficult to determine beforehand the number of generations needed to find near-optimum solutions. Thus, in the proposed scheme, the termination of the GA is chosen to occur when the repetitions of the same solution have reached a predetermined number. Consequently, the algorithm terminates when a specific chromosome has been appeared for a sufficiently large number of times.

4. Description of assembly under investigation

The robotic arm that has been manufactured for the requirements of the present study is approximately 600 mm in length, able to pick up small object of 0.3kg, has 6 degrees of freedom and is constructed mainly from 3mm aluminium sheet. The pivoting parts include six standard step motors in order to keep the design as simple as possible. Additionally, the robotic arm under investigation incorporates a controller which governs the arm movements and operations. A simple, scalable control system has been adopted to allow coordinated Cartesian control which offers expandability for future research. The idea behind the specific design was to investigate the possibility of developing a simple robotic arm capable of moving small objects that is

constructed by using exclusively aluminium bars and sheets as well as basic conventional machines during manufacture.

Figure 2 depicts the manufactured robotic arm that consists of a waist, a shoulder, an elbow, a wrist and a gripper which are connected to each other via metal links. These links are appropriately designed and machined so that they may offer stability, smooth motion and, if required, effective support for specific motors.

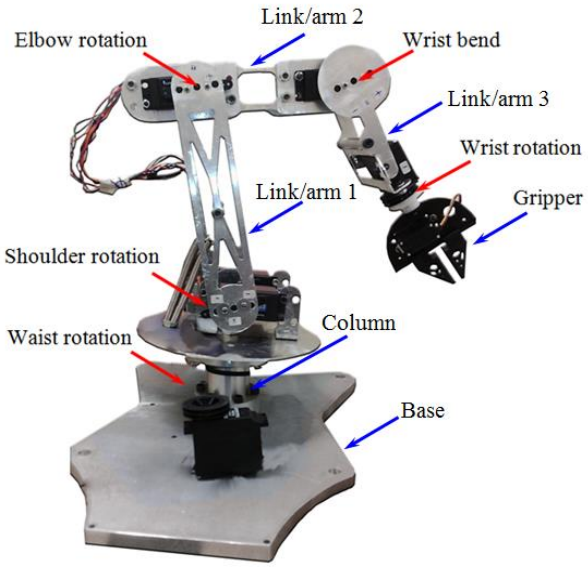

Figure 2. Robotic arm description.

The mechanical design of the robotic arm includes a heavy bottom base of 20mm thick aluminium which provides enough room for the more powerful motor which is responsible for the waist rotation. A column joined with a lighter upper base is manufactured to offer smooth shoulder rotation and appropriate motor support. The Link/arm 1 connects the shoulder and elbow while the link/arm 2 connects the elbow with the wrist. The second link is formed in such a way to bracket both the elbow rotation and wrist bend motors. The wrist rotation motor is attached on link/arm 3 which connects the wrist bend with the wrist rotation mechanisms. The gripper is

appropriately attached on the outer edge of this third link. All these robotic parts are coupled with appropriate cylindrical joints. The metal frame of the above described robotic arm is composed of several aluminium components which are illustrated and numbered in Figure 3.

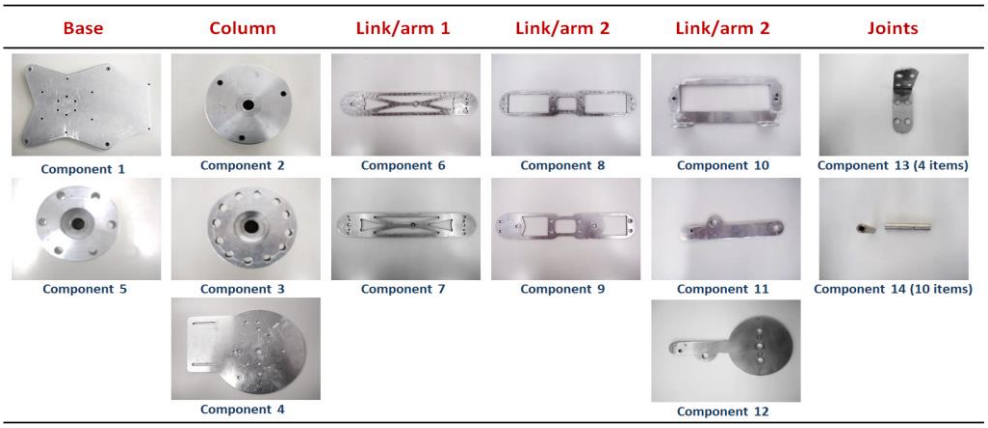

Figure 3. Numbered components of the robotic aluminium frame.

The metal components of the robotic frame have been machined using conventional processes in which appropriate pieces of raw material have been progressively cut and/or formed into the desired final shape and size. The whole machining process has been taken place in a simple assembly line of four workstations. The assembly line under consideration includes four types of machining operations, i.e. milling, drilling, lathing and bending. Conventional machines such as those depicted in Figure 4 have been utilized for the purpose of the present study.

Figure 4. The four types of machines of the assembly line.

5. Computational study

According to Figure 3, in order to construct the metal structure of the robotic arm 14 different type of components should be manufactured using the aforementioned machines. To be more precise, a total of number of 26 parts should be machined since the robotic arm design requires 4 and 10 items of component 13 and 14, respectively. Table 1 summarizes the fuzzy execution times for the 57 total tasks associated with the robot components. The computational study is focused on finding the optimum sequence of tasks assigned to stations which may lead to the minimum total fuzzy time for executing all tasks.

Considering fuzziness for the processing times, fuzzy data are represented by triangular fuzzy numbers. TFN is one of the most commonly used in the literature shapes of fuzzy numbers representation (Fonseca et al. 2005), composed of three estimates (the lowest expected, the most likely and the highest expected) of the unknown individual value. The reason of using triangular fuzzy shapes is because of

their computational simplicity in comparison with other fuzzy shapes, considering the calculations in Equations (1). TFNs differ from statistical distributions in the fact that they do not require historical data to establish their values. This is a major advantage of using TFNs as opposed to statistics.

In practice, this was achieved by assigning the 57 total tasks to 10 “workers” and writing down the resulting times. For all tasks, the most likely values for the TFNs (i.e. the fuzzy element with the membership value of 1) are considered to be the middle written times at the last column of Table 1. These most likely values for each task were set equal to the average of the corresponding ten measured times by the ten “workers”. The extreme TFN values in Table 1 were taken equal to the minimum and maximum times of the 10 execution times provided by the “workers”, respectively.

Table 1. Fuzzy execution times for each robot component.

No of tasks	**No of component**	**Type of task**	**Total fuzzy time** (s)
1	1	milling	(3096, 3105, 3111)
2	1	drilling	(73, 74, 76)
3	2	milling	(788, 792, 795)
4	2	drilling	(51, 53, 55)
5	3	milling	(822, 826, 831)
6	3	drilling	(40, 41, 42)
7	4	milling	(858, 863, 868)
8	4	drilling	(66, 67, 69)
9	5	milling	(518, 523, 527)
10	5	drilling	(44, 45, 46)
11	6	milling	(842, 849, 854)
12	6	drilling	(42, 43, 45)
13	7	milling	(1043, 1052, 1058)
14	7	drilling	(17, 18, 19)
15	8	milling	(780, 785, 789)
16	8	drilling	(57, 59, 61)
17	9	milling	(803, 806, 810)
18	9	drilling	(54, 55, 57)
19	10	milling	(540, 544, 549)
20	10	drilling	(24, 25, 26)
21	10	bending	(40, 41, 42)
22	11	milling	(390, 396, 401)
23	11	drilling	(25, 26, 27)
24	12	milling	(381, 385, 390)
25	12	drilling	(40, 41, 42)
26	13 (1 out of 4 items)	milling	(947, 951, 956)
27	13	drilling	(248, 252, 255)
28	13	bending	(39, 41, 42)
29-37	13 (3 items)		
38	14 (1 out of 10 items)	lathing	(1560, 1569, 1576)
39	14	drilling	(63, 65, 66)
40-57	14 (9 items)		

The precedence constraints are summarized in the following:

- task1 precedes task2
- task3 precedes task4
- task5 precedes task6
- task7 precedes task8
- task9 precedes task10
- task11 precedes task12
- task13 precedes task14
- task15 precedes task16
- task17 precedes task18
- task19 precedes task20 and task20 precedes task21
- task22 precedes task23
- task24 precedes task25
- task26 precedes task27 and task27 precedes task28
- task29 precedes task30 and task30 precedes task31
- task32 precedes task33 and task33 precedes task34
- task35 precedes task36 and task37 precedes task38
- task38 precedes task39
- task40 precedes task41
- task42 precedes task43
- task44 precedes task45
- task46 precedes task47
- task48 precedes task49
- task50 precedes task51
- task50 precedes task51

- task52 precedes task53
- task54 precedes task55
- task56 precedes task57

The minimum fuzzy cycle time is yielded considering the total fuzzy processing time for each of the 57 tasks (see Table 1) as well precedence relations given above. The resulted minimum fuzzy cycle time after running the proposed GA is (8150, 8211, 8259) s. Concerning the station load for each workstation, it is provided by the GA best solution considering the aforementioned constraints and involves the task sequence assigned to each workstation. In particular, Workstation1 starts with task 53 and after finishing, it continues with task 24 and so on until it finishes with task 3. Similarly, Workstation2 starts with task 22, then accomplishes task 38 and finishes with task 10. The robotic arm is manufactured when all tasks assigned to each workstation have been accomplished. For the problem addressed here, the resulting task sequence and the fuzzy station time for each workstation are presented in Table 2. The simulation test was implemented in Matlab and run on a Pentium IV 2.13 GHz core2 PC and the CPU time was 993 sec.

Table 2. Results provided by the proposed GA.

Workstation z	**Task sequence (** S_z **)**	**Fuzzy station time (** $\tilde{tS}_z$ **)** (s)
1	52-53-24-40-35-17-5-6-29-30-3	(8159, 8207, 8253)
2	22-38-39-31-4-25-15-16-11-19-20-21-23-12-1-9-10	(8151, 8210, 8259)
3	18-42-43-48-49-56-57-32-33-34-50-51-2-36-37	(8140, 8202, 8251)
4	26-27-54-55-44-45-28-13-14-46-47-7-8-41	(8150, 8211, 8259)
Fuzzy cycle $\tilde{C}$ (s)	(8150, 8211, 8259)	

Figure 5 depicts the triangular membership function for the fuzzy cycle time. It is clear that the least likely values are 8150 and 8259 while it is found that the most likely value is 8211.

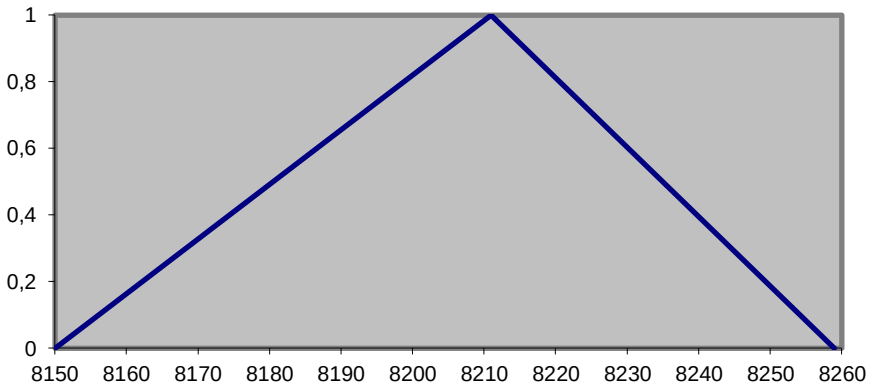

Figure 5. The fuzzy cycle time $\tilde{c}$.

To compare the fuzzy numbers, the greatest associate ordinary number (see Equation (2)) is used. Thus, the value for the cycle time presented in the following has been calculated through Equation (2), Figure 6, 7 and 8 illustrate the evolution of the GA through the generations for the best, average and worst individuals, respectively. It is clear from Figure 6 that the best cycle time (equal to 8207.75 s corresponding to (8150, 8211, 8259) s) is achieved in the 1459th generation.

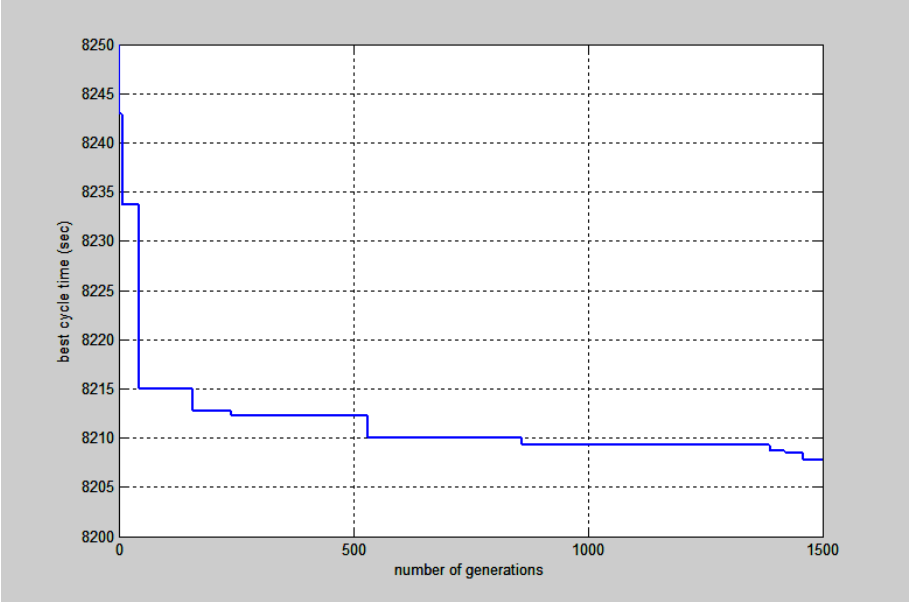

Figure 6. Best cycle time versus generations.

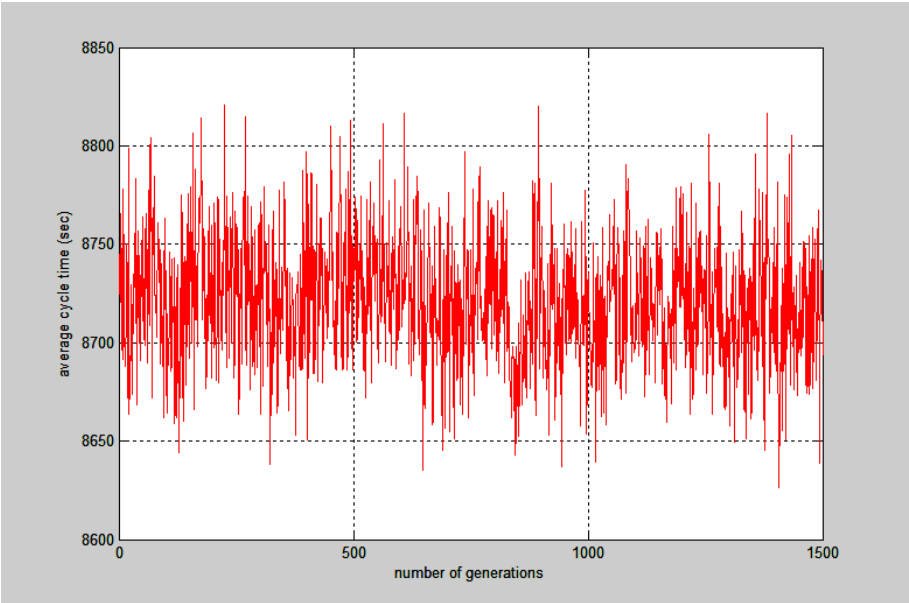

Figure 7. Average cycle time versus generations.

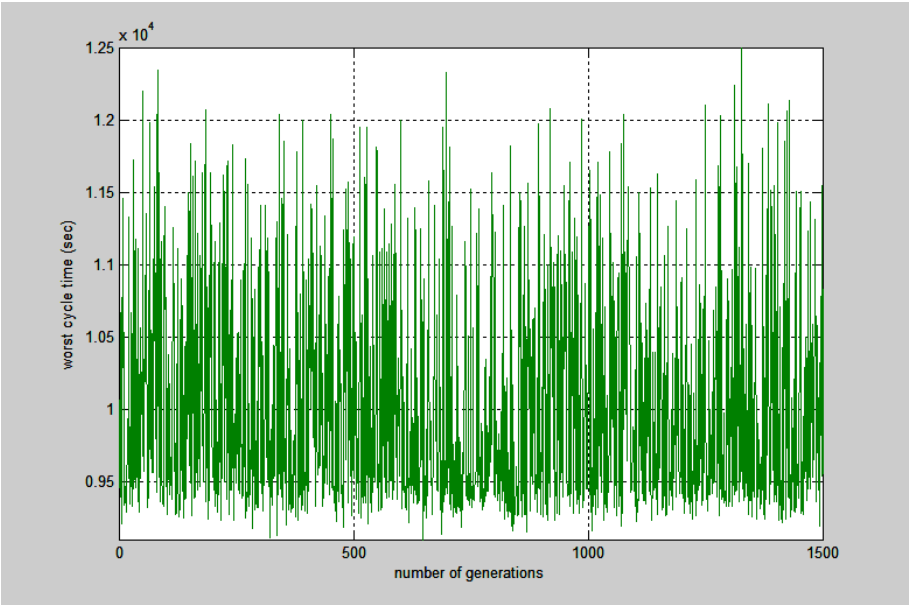

Figure 8. Worst cycle time versus generations.

6. Conclusions

Due to the fact that the nature of manufacturing systems is accompanied with uncertainty, the main idea for this paper is to treat the problem considering fuzziness in the operation times. Thus, the main focus of this work lies on the construction process considering variability and ambiguity associated with real situations. In this context, this paper studies the solution of the fuzzy assembly line balancing problem type-2 for the real problem of constructing a robotic arm. The robot components are formed using a number of machine tools, which are handled by a specific number of "workers". The metal parts of the robotic arm are manufactured in four machining workstations. The construction process is enhanced in terms of time reduction using a GA that takes into account fuzziness in times and is subject to the constraints imposed by the precedence relations.

The proposed approach was tested in a real manufacturing environment, where real data was yielded for the simulation tests. The experimental results demonstrated

that the approach is effective and efficient at determining the optimum fuzzy cycle time for the accomplishment of robotic arm construction without violating the precedence constraints. Considering future research, a U-shaped assembly line layout could be applied, where stations work at two segments of the line facing each other simultaneously.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: ARCHIMEDES III. Investing in knowledge society through the European Social Fund.

References

Ariffin, M. K. A., Fathi M., and Ismail N. 2012. "A new heuristic method to solve straight assembly line balancing problem." *Pertanika Journal of Science & Technology* 20 (2): 355–369.

Atasagun, Y., and Kara Y. 2014. "Bacterial foraging optimization algorithm for assembly line balancing." *Neural Computing & Applications* 25: 237–250.

Avikal, S., Mishra P. K., and Jain R. 2012. "A model for assembly line balancing problems." Proceedings of the 2012 Students Conference on Engineering and Systems (SCES 2012), 6199117, 1–4.

Blum, C. 2011. “Iterative beam search for simple assembly line balancing with a fixed number of work stations.” *SORT-Statistics and Operations Research Transactions* 35 (2): 145–164.

Dou, J., Li J, and Lv Q. 2011. “A hybrid particle swarm algorithm for assembly line balancing problem of type 1.” Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation (ICMA 2011), 5986373, 1664–1669.

Dou, J., Li J., and Su C. 2013. “A novel feasible task sequence-oriented discrete particle swarm algorithm for simple assembly line balancing problem of type 1.” *International Journal of Advanced Manufacturing Technology* 69 (9–12): 2445–2457.

Dou, J., Su C., and Li J. 2011. “A discrete particle swarm optimization algorithm for assembly line balancing problem of type 1.” Proceedings of the 3rd International Conference on Measuring Technology and Mechatronics Automation (ICMTMA 2011), 1, 5720697, 44–47.

Fathi, M., Jahan A., Ariffin M. K. A., and Ismail N. 2011. “A new heuristic method based on CPM in SALBP.” *Journal of Industrial Engineering International* 7 (13): 1–11.

Fonseca, D. J., Guest C. L., Elam M., and Karr C. L. 2005. “A fuzzy logic approach to assembly line balancing.” *Mathware & Soft Computing* 12: 57–74.

Gen, M., Tsujimura Y., and Li Y. 1996. “Fuzzy assembly line balancing using genetic algorithms.” *Computers and Industrial Engineering* 31 (3–4): 631–634.

Goldberg, D. E. 1989. “Genetic algorithms in search, optimization and machine learning.” Addison-Wesley, Reading, Mass.

Goncalves, J. F., and Almeida J. R. D. 2002. "A hybrid genetic algorithm for assembly line balancing." *Journal of Heuristics* 8 (6): 629–642.

Gu, L., Hennequin S., Sava A., and Xie X. 2007. "Assembly line balancing problems solved by estimation of distribution." Proceedings of the 3rd IEEE International Conference on Automation Science and Engineering (IEEE CASE 2007), 4341810, 123–127.

Holland, J. H. 1975. "Adaption in natural and artificial systems." The University of Michigan Press, Ann Harbor, MI.

Kalender, F. Y., Yilmaz M. M., and Turkbey O. 2008. "A fuzzy approach to assembly line balancing problem." *Journal of the Faculty of Engineering and Architecture of Gazi University* 23 (1): 129–138.

Kaufmann, A., and Gupta M. M. 1985. "Introduction to Fuzzy Arithmetic." Van Nostrand Reinhold.

Khoshalhan, F., and Zegordi S. H. 2003. "Solving type one and type two fuzzy assembly line balancing problems using genetic algorithms." *Amirkabir Journal of Science and Technology* 14 (55 D): 910–923.

Kilincci, O. 2010. "A Petri net-based heuristic for simple assembly line balancing problem of type 2." *International Journal of Advanced Manufacturing Technology* 46 (1–4): 329–338.

Kilincci, O., and Bayhan G. M. 2006. "A Petri net approach for simple assembly line balancing problems." *International Journal of Advanced Manufacturing Technology* 30 (11–12): 1165–1173.

Kilincci, O., and Bayhan G. M. 2008. "A *P*-invariant-based algorithm for simple assembly line balancing problem of type-1." *International Journal of Advanced Manufacturing Technology* 37 (3–4): 400–409.

Kim, Y. K., Kim Y. J., and Kim Y. 1996. “Genetic algorithms for assembly line balancing with various objectives.” *Computers & Industrial Engineering* 30 (3): 397–409.

La Scalia, G., Micale R., Aiello G., and Enea M. 2013. “Solving type-2 assembly line balancing problem with fuzzy binary linear programming.” *Journal of Intelligent and Fuzzy Systems* 25 (3): 517–524.

Lapierre, S. D., Ruiz A., and Soriano P. 2006. “Balancing assembly lines with tabu search.” *European Journal of Operational Research* 168(3): 826–837.

Michalewicz, Z. 1996. “Genetic algorithms + data structures = Evolution program,” 3rd edn. Springer, Berlin.

Mutlu, O., and Ozgormus E. 2012. “A fuzzy assembly line balancing problem with physical workload constraints.” *International Journal of Production Research* 50 (18): 5281–5291.

Nearchou AC. Multi-objective balancing of assembly lines by population heuristics. International Journal of Production Research 2008;46/8:2275–97.

Ozcan, U., and Toklu B. 2009. “Multiple-criteria decision-making in two-sided assembly line balancing: A goal programming and a fuzzy goal programming models.” *Computers and Operations Research* 36 (6): 1955–1965.

Rajabalipour Cheshmehgaz, H., Haron H., Kazemipour F., and Desa M. I. 2012. “Accumulated risk of body postures in assembly line balancing problem and modeling through a multi-criteria fuzzy-genetic algorithm.” *Computers and Industrial Engineering* 63 (2): 503–512.

Scholl, A., and C. Becker. 2006. “State-of-the-art exact and heuristic solution procedures for simple assembly line balancing.” *European Journal of Operational Research* 168 (3): 666–693.

Sulaiman, M. N. I., Choo Y. H., and Chong K. E. 2011. “Ant colony optimization with look forward ant in solving assembly line balancing problem.” Proceedings of 2011 Conference on Data Mining and Optimization (3rd DMO 2011), 5976514, 115–121.

Tang, Q., Lu S., Li M., and Floudas C. A. 2011. “Novel cellular automata algorithm for assembly line balancing problem of type-2.” Proceedings of the 2011 6th International Conference on Pervasive Computing and Applications (ICPCA 2011), 6106542, 422–428.

Tapkan, P., Ozbakir L., and Baykasoglu A. 2012. “A Bees Algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem.” *Optimization Letters* 6 (6): 1039–1049.

Tsujimura, Y., Gen M., and Kubota E. 1995. “Solving fuzzy assembly-line balancing problem with genetic algorithms.” *Computers and Industrial Engineering* 29 (1–4): 543–547.

Yeh, D. H., and Kao H. H. 2009. “A new bidirectional heuristic for the assembly line balancing problem.” *Computers & Industrial Engineering* 57 (4): 1155–1160.

Zacharia, P. T., and Nearchou A. C. 2012. “Multi-objective fuzzy assembly line balancing using genetic algorithms.” *Journal of Intelligent Manufacturing* 23 (3): 615–627.

Zadeh, L. A. 1965. “Fuzzy sets.” *Information and Control* 8 (3): 338–358.

Zhang, R., Chen D., Wang Y., Yang Z., and Wang X. 2007. “Study on line balancing problem based on improved genetic algorithms.” Proceedings of the 2007 International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2007), 4340283, 2033–2036.

Zhang, Z. Q., Cheng W. M., Tang L. S., and Zhong B. 2007. “Ant algorithm with summation rules for assembly line balancing problem.” Proceedings of 2007 International Conference on Management Science and Engineering (14th ICMSE 2007), 4421875, 369–374.

Zheng, Q., Li M., Li Y., and Tang Q. 2013. Station ant colony optimization for the type 2 assembly line balancing problem.” *International Journal of Advanced Manufacturing Technology* 66 (9–12): 1859–1870.