Mathware & Soft Computing 12 (2005), 57-74

A Fuzzy Logic Approach to Assembly Line Balancing

D.J. Fonseca[1], C.L. Guest[1], M. Elam[1], and C.L. Karr[2]

[1] Department of Industrial Engineering
[2]Department of Aerospace Engineering and Mechanics

University of Alabama

dfonseca@coe.eng.ua.edu, cguest@ups.com
melam@ceo.eng.ua.edu, ckarr@coe.eng.ua.edu

Abstract

This paper deals with the use of fuzzy set theory as a viable alternative method for modelling and solving the stochastic assembly line balancing problem. Variability and uncertainty in the assembly line balancing problem has traditionally been modelled through the use of statistical distributions. This may not be feasible in cases where no historical data exists. Fuzzy set theory allows for the consideration of the ambiguity involved in assigning processing and cycle times and the uncertainty contained within such time variables. Two widely used line balancing methods, the COMSOAL and Ranked Positional Weighting Technique, were modified to solve the balancing problem with a fuzzy representation of the time variables. The paper shows that the new fuzzy methods are capable of producing solutions similar to, and in some cases better than, those reached by the traditional methods.

Keywords: Line Balancing, Fuzzy Sets, COMSOAL, Ranked Positional Weighting Technique.

1 Introduction

The manufacturing assembly line was first introduced by Henry Ford in the early 1900's. It was designed to be an efficient, highly productive way of manufacturing a particular product. The basic assembly line consists of a set of workstations arranged in a linear fashion, with each station connected by a material handling device. The basic movement of material through an assembly line begins with a part being fed into the first station at a predetermined feed rate. A station is considered any point

on the assembly line in which a task is performed on the part. These tasks can be performed by machinery, robots, and/or human operators. Once the part enters a station, a task is then performed on the part, and the part is fed to the next operation. The time it takes to complete a task at each operation is known as the process time [14].

The cycle time of an assembly line is predetermined by a desired production rate. This production rate is set so that the desired amount of end product is produced within a certain time period [3]. For instance, the production rate might be set at 480 parts per day. Assuming an eight-hour shift, this translates into a requirement of 60 parts per hour (1 part per minute) being produced by the assembly line. In order for the assembly line to maintain a certain production rate, the sum of the processing times at each station must not exceed the stations' cycle time. If the sum of the processing times within a station is less than the cycle time, idle time is said to be present at that station [5].

One of the main issues concerning the development of an assembly line is how to arrange the tasks to be performed. This arrangement may be somewhat subjective, but has to be dictated by implied rules set forth by the production sequence [10]. For the manufacturing of any item, there are some sequences of tasks that must be followed.

The assembly line balancing problem (ALBP) originated with the invention of the assembly line. Helgeson et. al [8] were the first to propose the ALBP, and Salveson [13] was the first to publish the problem in its mathematical form. However, during the first forty years of the assembly line's existence, only trial-and-error methods were used to balance the lines [5]. Since then, there have been numerous methods developed to solve the different forms of the ALBP.

Salveson [13] provided the first mathematical attempt by solving the problem as a linear program. Gutjahr and Nemhauser [7] showed that the ALBP problem falls into the class of NP-hard combinatorial optimization problems. This means that an optimal solution is not guaranteed for problems of significant size. Therefore, heuristic methods have become the most popular techniques for solving the problem.

Even though the assembly line balancing problem has received significant attention over its lifetime, many companies still do not utilize the methods proposed in the literature. This fact can be seen in a survey conducted by Chase in 1974. His survey showed that roughly only 5% of companies with production lines utilize traditional line balancing techniques to balance their assembly lines [4]. A more recent article by Milas [12] showed that this trend is still valid in today's manufacturing environment. Milas [12] found that most companies perform their line balancing based on historical precedent or the 'gut feel' of their engineers.

Evidently, the lack of industrial use of conventional line balancing techniques suggests that there may be some major drawbacks to the use of current techniques. One possibility is that current methods are not adequate and/or flexible enough to model real situations of the assembly line environment, or that designers are just not familiar with the published ALBP literature [5].

2 The ALBP: A Numerical Example

To illustrate the dynamics involved in balancing a production line, a numerical example of the ALBP is presented here. The objective is to group the individual tasks into workstations. The demand will be set at an arbitrary value of 20 units per hour. This is equal to a cycle time of 0.05 hours per unit. The task times and precedence relationships can be seen in Figure 1.

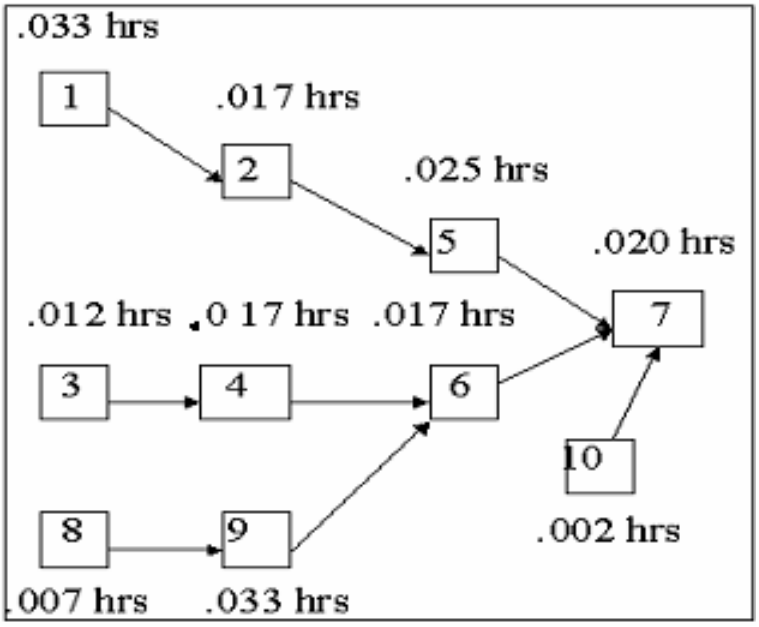

Figure1: ALBP Example Task Times and Precedence Diagram

The total work time involved for all 10 tasks is 0.183 hours. With a cycle time of 0.05 hours, there should be approximately four workstations (0.183 / 0.05 = 3.66 ~

4). The solution begins by grouping tasks 1 and 2 together into Workstation 1. The sum of their task times equals 0.050 hours, which leaves zero idle time at this workstation. Therefore, Workstation 2 is opened and tasks 3, 4, and 6 are assigned to it. Their process times total 0.046 hours, which leaves a total of 0.004 hours of idle time at that station. No more tasks will fit at workstation 2, so a new workstation must be opened. Tasks 8 and 9 are assigned to Workstation 3, while tasks 5, 7, and 10 are assigned to Workstation 4. Figure 2 shows the final grouping of tasks into their workstations.

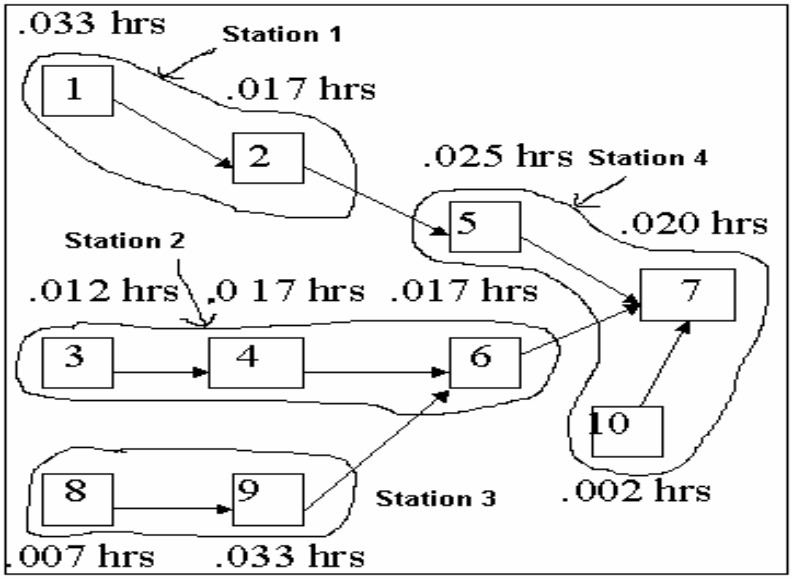

Figure 2: Final Workstations for ALBP Example

The idle time for each station is found by subtracting the total time from the given cycle time. By using Equation 1, the percentage idle time for the whole production line can be found.

$$\text{Idle \%} = (S (C - Ti)) / (n * C) * 100 \quad (1)$$

Where C is the cycle time, Ti is the total time for workstation i, and n is the total number of workstations. For this example, the percentage idle time is as follows:

Idle % = [(0.05-0.05) + (0.05-0.046) + (0.05-0.04) + (0.05-0.047)] / (4*0.05)= 85%

3 The COMSOAL Technique

One of the methods most widely used to solve the ALBP is the COMSOAL technique, developed by Arcus in 1966. The technique uses a basic record-keeping scheme to solve the ALBP. The approach allows for a large number of sequences to be generated and compared against each other. The sequences are developed by randomly choosing an available task, and assigning it to the next feasible workstation. If the chosen task can not fit in the current workstation due to the length of its processing, a new workstation is then opened. The technique keeps track of all relevant workstation attributes, such as the remaining workstation idle time and the tasks to be assigned. Tasks must satisfy all constraints in order to be considered for assignment [2].

The COMSOAL approach compares generated sequences based on their number of workstations. Every time a new sequence is developed, the number of workstations in that sequence is stored. A new sequence is then generated, and the number of workstations is compared against the current best solution (i.e. the upper bound solution). If the number of workstations is higher than the upper bound, the sequence is discarded and a new solution is developed. If it is lower than the current upper bound, the sequence is saved and the upper bound is updated [2].

This computerized approach to solving the ALBP has several advantages to other techniques. The heuristic is fairly easy to program, which allows for large problems to be modeled in a reasonable amount of time. Also, feasible solutions are quickly found, and the amount of computational effort used is directly proportional to the quality of the generated solution. Lastly, the method can be applied to a wide array of decision problems. As long as the solutions can be built sequentially, and an evaluation function can be developed to compare solutions, the COMSOAL technique can be applied to solve the problem.

4 The Ranked Positional Weighting Technique (RPWT)

This is another widely used line balancing method developed by Helgeson and Birnie [8]. Unlike the COMSOAL technique, it is a single-pass heuristic where only one feasible solution is generated. It is a greedy heuristic because it picks tasks based on a previously ordered list and does not look ahead into the future. In this method, tasks are prioritized based on the cumulative sum of their individual processing times, and the processing times of the job's successors. This sum is called the 'positional weight' (PW(i)) of the task, and the tasks with higher positional weights are assigned first. Tasks are assigned to the current open workstation if permitted by their processing time. Otherwise, a new workstation is opened. The idea behind the heuristic is that the greater the number of tasks that are available for assignment, the

higher the probability that at least one task will fit at a particular workstation. By following this logic, it is possible to use the most of each station's cycle time by finding appropriate tasks. This should result in a fewer required stations with minimized associated idle times [2].

The procedure for performing the RPWT can be seen in the following steps provided by Askin and Standridge [2]:

1. Task ordering: For all tasks i = 1,….,N, compute PW(i). Order (rank) tasks by nonincreasing PW(i).
2. Task assignment: For ranked tasks i = 1,….,N, assign task i to the first feasible workstation.

By examining the fundamentals of the heuristic, it is easy to see that the positional weight of a task will assure that its predecessors have already been assigned. If tasks are assigned to workstations that are at least as large (numerically) as their immediate predecessor, all precedence relations will then be implicitly satisfied. In order to further demonstrate the RPWT, a numerical example is presented as follows.

4 Fuzzyfication of the COMSOAL and RPWT

In order to develop fuzzy versions of the COMSOAL and RPWT methods, cycle and processing times had to be represented as fuzzy numbers. Due to the nature of the processing and cycle times, the most appropriate fuzzy sets to use in depicting these values are triangular fuzzy numbers (TFN's). Instead of assigning one particular value for the time variables, as in the case of deterministic methods, TFN's establish extreme points to represent the most likely and least likely values for the individual variables. The use of TFN's to model time values allows the practitioner to account for variability and ambiguity, similar to the use of statistical distributions. However, TFN's differ from statistical distributions in the fact that they do not require prior knowledge or historical data to establish their values. This is one major advantage of using TFN's as opposed to statistics. This was done by defining the existing deterministic time as the most likely value for the new TFN. This implies that the deterministic value represents the fuzzy element in the set with the membership value of 1. In order to get the least likely values for the new TFN's, with the membership values of 0, a value of 1 was added and subtracted from the most likely values. This same procedure was performed to adapt the deterministic cycle times into the fuzzy domain.

4.1 Fuzzification of COMSOAL

The COMSOAL heuristic solves the assembly line balancing problem by generating random sequences for the ordering of tasks. After each random sequence is

generated, the number of workstations and associated idle times are stored for comparison with the last sequence. If the current sequence is more efficient (i.e. has less number of workstations and smaller idle times) than the previous one, it is stored as the new lower bound. This procedure is continued for a pre-established number of iterations, and the random sequence with the highest efficiency is kept as the current optimum solution.

In order to adapt this heuristic to handle fuzzy processing and cycle times, there were no major changes that had to be performed. The basic functioning of the COMSOAL method was suitable for operating in the fuzzy domain, but the intermediate steps within the random sequence generation had to be modified. The use of the existing precedence constraint diagrams is still needed, and the violation constraints must still be avoided. However, the time variables are now represented as triangular fuzzy numbers, which required some intermediate steps to be adapted to handle such type of numbers.

The most predominant change to the existing crisp method deals with how the processing times are accumulated for each task that has been assigned. In the current heuristic, when a task is assigned to a workstation, its associated processing time is subtracted from the remaining idle time of the workstation. When the task chosen to be assigned next is selected, its processing time must be less than remaining workstation's idle time. Otherwise, a new workstation must be opened to accommodate this task. In the adapted fuzzy heuristic, the tasks' fuzzy processing times, 3?, are accumulated using the fuzzy addition operator. When the subsequent task is chosen for assignment to the current workstation, it may only be assigned if its processing time will not cause the workstation's total task time to exceed the fuzzy cycle time, &?max. In other words, the total fuzzy task time for a workstation, W? is the cumulative sum of all the processing times of the tasks assigned to that station. A new task may only be assigned to that station if the addition of its processing time to the current total task time will not cause the total task time to exceed the permitted cycle time. Otherwise, a new workstation must be opened to accommodate that task. In order to compare the total task time of a workstation to its permitted cycle time, the Mean Comparison Method was utilized. Overall, the main difference between the crisp and fuzzy heuristics lies in how new processing times are accumulated. In the crisp version, they are subtracted from the remaining idle time, while in the fuzzy version they are added to the total task time.

The other steps in the new fuzzy heuristic are very similar to the existing crisp method, with slight differences in mathematical calculations. Tasks are selected for assignment based on precedence relations and the generation of a random number. The fuzzy heuristic continues until all tasks have been assigned to their workstations. After a solution is generated, the number of workstations and the associated idle times are calculated. To compute the idle time at a workstation, the fuzzy

subtraction operator is utilized. Once the idle times for each of the workstations are calculated, the fuzzy efficiency and fuzzy idle percentage of the line are established. The fuzzy line efficiency is then stored as the lower bound for the line balancing solution. A new solution sequence is then generated and the entire process is repeated. Once a certain number of iterations have been performed, the current optimal solution is selected by comparing the line efficiencies and number of workstations. The sequence with the lowest number of workstations and highest line efficiency is then chosen as the current best solution.

The notation involved in the constructed fuzzy COMSOAL to solve the stochastic assembly line balancing problem is described in Table 1.

$\tilde{P}_j$	fuzzy processing time of job J_j, $j = 1,2,\ldots,n$, defined as a triplet (a1, a2, a3)
$\tilde{t}_i$	fuzzy triangular time required to complete all jobs assigned to workstation W_i, $i = 1,2,\ldots,m$
$\tilde{c}$	fuzzy cycle time ($\max \tilde{t}_i$) after assignments
$\tilde{I}_i$	fuzzy idle time for workstation W_i, $i = 1,2,\ldots,m$ ($\tilde{C}_{max} - \tilde{t}_i$)
$\tilde{T}$	total processing time in all work stations ($\Sigma \tilde{t}_i$)
$\tilde{C}_{max}$	permitted fuzzy cycle time
$\tilde{D}$	fuzzy balance efficiency = $\tilde{T}$ / (m * $\tilde{c}$)

Table 1. Notation for Fuzzy COMSOAL Procedure

The method used to compare W + 3? to &?max is called the Mean Comparison Method (Gen et. al, 1996). It compares two TFN's by defining W+ 3? = - = (a1, a2, a3) and &?max = %?= (b1, b2, b3); we say that - > %? IFF a2>b2. If - > %? then we can add job 3? to the current workstation without exceeding the cycle time. If a2<b2, then - < %? and we can not assign job 3? to the current workstation.

The fuzzy COMSOAL heuristic involves the following eleven steps:

1) Establish each job's processing time and given cycle times as TFN's
2) Establish precedence relationships
3) Begin assigning first chosen task to Workstation 1, and accumulate the fuzzy process times by using fuzzy addition: [- + %?= (a1+b1, a2+b2, a3+b3) = W]. Tasks will be chosen based on precedence relations and random number generation, as in the original COMSOAL technique

4) Continue to assign job j to current workstation 1, as long as: Wi + tj <= cmax (Use Mean Comparison Method)
5) Open a new workstation if Wi exceeds cmax with the addition of new job j
6) Continue until all jobs are assigned
7) Calculate remaining idle time at each station after final assignment by using fuzzy subtraction: [Ii = cmax - Wi = (a1-b3, a2-b2, a3-b1)]
8) Determine Fuzzy(max Wi) and Fuzzy(S Wi)
9) Compute Fuzzy Eff = Fuzzy (m * Fuzzy)
10) Compute Idle %: Idle % = (S(cmax - Wi)) / (m * cmax) * 100
11) Generate a new sequence using the same method. The new sequence may differ from existing one due to the change in random numbers. The two sequences are compared based on their efficiency. If the new efficiency (new sequence) is higher than the current efficiency (current sequence), then the current sequence is kept as the lower bound. If not, the new sequence is discarded and the current sequence remains as the lower bound

The process is repeated until the highest efficiency possible is reached.

4.2 Fuzzification of RPWT

The Ranked Positional Weighting Technique is a single-pass heuristic that generates only one solution. The solution is derived by assigning positional weights to each of the tasks based on the sum of their processing times, and the processing times of their successors. Tasks are then ranked in decreasing order of their positional weights, and are assigned to workstations based on this ranked order. The major idea behind this heuristic is that tasks with a larger number of successors have higher positional weights, and therefore, will be assigned to upstream workstations.

As in the case of the COMSOAL heuristic, the underlying methodology of the procedure does not change when adapted to handle fuzzy processing and cycle times. Positional weights are calculated for the tasks, and precedence constraints must not be violated when assigning these tasks. However, in order for the RPWT method to be effective in the new fuzzy environment, some of the intermediate steps are adapted to handle the fuzzy representation of time variables.

The first major adaptation to the crisp heuristic relates to the calculation of the positional weights for each task. In the existing crisp heuristic, positional weights are calculated by summing the processing time of the current task with the processing times of all of its successors. This is done using standard mathematical addition. In the new fuzzy heuristic, the fuzzy positional weight for each task, Fuzzy PW, is also calculated by summing the processing time of the current task with the processing times of all its successors. However, since the processing times are now

represented by triangular fuzzy numbers, the fuzzy addition operator has to be utilized in this calculation. The next adaptation of the crisp heuristic is in relation to the ranking of tasks based on their positional weights. The traditional method ranks these tasks in decreasing order by comparing the values of the crisp positional weights. This step is also performed in the new fuzzy heuristic, but a ranking method for triangular fuzzy numbers is adopted. The fuzzy ranking method used in the new heuristic is known as the Average Height Method. This method is adopted from a paper by Hong and Chuang [9]. Basically, the method calculates the average height of the triangular fuzzy number by summing the three triangular values and dividing this sum by 3. This results in a crisp number that is easier to rank. This step is performed on the fuzzy positional weight for each task, and then the tasks are ranked in a decreasing order based on their average height.

The last adaptation deals with how task processing times are handled when a new task is added to a workstation. As in the case of the traditional COMSOAL technique, when a task is assigned to a workstation, the existing RPWT method subtracts the associated crisp processing time from the remaining idle time at that workstation. In the fuzzy RPWT method, fuzzy processing times are accumulated using the fuzzy addition operator whenever a new task is added to a workstation. However, before the chosen task can be assigned to the current workstation, it must be shown that the addition of this task will not violate the permitted cycle time of the workstation. As in the fuzzy COMSOAL technique, this validation is performed by using the Mean Comparison Method. If the addition of the chosen task violates the permitted cycle time, then a new workstation must be opened to accommodate the task. However, in the RPWT technique, the next subsequent task in the ranked positional weight list may be assigned if the current task time does not fit. This can be done only if the precedence constraints are not violated.

The remaining steps in the fuzzy RPWT coincide with those in the crisp RPWT. Tasks are assigned until a single solution is generated, and this solution is chosen as the current optimal solution. Some other final calculations are included in the fuzzy RPWT method, but are not necessary for generating a solution sequence. These calculations are the fuzzy line efficiency and fuzzy idle percentage of the balanced line. The notation involved in the fuzzy RPWT heuristic to solve the stochastic assembly line balancing problem is depicted in Table 2.

The method used to compare W + 3 to &max is called the Mean Comparison Method [6]. It compares two TFN's by defining W+ 3 = - = (a1, a2, a3) and &max = %= (b1, b2, b3); we say that - > %IFF a2>b2. If - > %then we can add job 3 to the current workstation without exceeding the cycle time. If a2<b2, then - < %and we can not assign job 3 to the current workstation. The fuzzy RPWT procedure involves the following twelve steps:

1) Establish each job's processing time and given cycles time as TFN's
2) Establish precedence relationships
3) Compute 3? ?for each task by using fuzzy addition:
   [- + %?= (a1+b1, a2+b2, a3+b3)]
   3? ?= 3?+ S 3?(summation for all successors of job j)

$\tilde{P}_j$	fuzzy processing time of job J_j, $j = 1,2,\ldots,n$, defined as a triplet (a1, a2, a3)
$\tilde{P}_r$	fuzzy processing time for job r, in which job r is a successor of job j
$\tilde{P}\tilde{W}_i$	fuzzy Ranked Positional Weight
$\tilde{t}_i$	fuzzy time required to complete all jobs assigned to workstation W_i, $i = 1,2,\ldots,m$
$\tilde{c}$	fuzzy cycle time ($\max \tilde{t}_i$) after assignments
$\tilde{I}_i$	fuzzy idle time for workstation W_i, $i = 1,2,\ldots,m$ ($\tilde{C}_{max} - \tilde{t}_i$)
$\tilde{T}$	total processing time in all work stations ($\Sigma \tilde{t}_i$)
$\tilde{C}_{max}$	permitted fuzzy cycle time
$\tilde{D}$	fuzzy balance efficiency = $\tilde{T}$ / (m * $\tilde{c}$)

Table 2. Notation for Fuzzy RPWT Procedure

4) Put tasks in descending order based on their 3? ?into List A. To rank the tasks for descending order, use the Average Height Method: Compute H for task - and %?where H(-) = (1/3)*(a1+a2+a3) and H(%?=(1/3)*(b1+b2+b3). We can say that - > %?if H(-) > H(%?. Otherwise, we assume that - is not > %?
5) Begin assigning the first task in List A to Workstation 1, and accumulate the fuzzy process times by using fuzzy addition:
   [- + %?= (a1+b1, a2+b2, a3+b3) = W?]
   Tasks will be chosen based on precedence relations and their 3? ?rank
6) Continue to assign job j to current workstation 1, as long as:
   W? + 3?<= &?max (Use Mean Comparison Method)
7) Open a new workstation if W?exceeds &?max with the addition of new job j
8) Continue until all jobs are assigned
9) Calculate remaining idle time at each station after final assignment by using fuzzy subtraction: [?i = &?max -Wi = (a1-b3, a2-b2, a3-b1)]
10) Determine F?(max W?) and 7 ?(S W?)
11) Compute ? ?= 7 ?/ (m * F?)

12) Compute Idle %: Idle % = (S(หmax - Wi)) / (m * หmax) * 100

5. Validation of the Fuzzy Heuristics

In order to validate the developed fuzzy heuristics, example problems from the available literature were used as test data. Two of the problems, involving crisp data sets, were adapted by transforming their time variables into fuzzy ones. The third example was taken from a study by Tsujimura et. al [15], which already represented the processing and cycle times as triangular fuzzy numbers.
The criteria used to compare the original solutions to those developed by the new heuristics were based on the number of workstations, their respective idle times, and the efficiency of the balanced line. However, since two of the test data sets were originally from the crisp domain, the values for the idle times and line efficiencies were also crisp.

In order to efficiently perform the validation phase of this research, the fuzzy COMSOAL and RPWT techniques were automated using Visual Basic (VB).

5.1 Validation of the Fuzzy COMSOAL

The new fuzzy COMSOAL heuristic was validated through three example problems from the literature. The solutions reached by the new methods were then compared to the solutions generated from the original heuristics.

The first test problem was drawn from Askin and Standridge [2]. The problem data set originally contained crisp values, so its time variables were transformed into triangular fuzzy numbers. The solutions generated from the original and fuzzy COMSOAL are shown in Table 3 and 4, respectively. These tables also show the line efficiency and idle percentage values for the two solutions.

Workstation	Assigned Task	Idle Time (sec)
1	4, 6, 5	6
2	1, 2, 3, 7, 8, 10	9
3	9, 11	9
4	12	54
Line Efficiency = 0.789063		
Idle Percentage = 27.85714		

Table 3. Results from Original Crisp Solution Method – Test Problem 1

Workstation	Assigned Task	Fuzzy Idle Time (sec)
1	5, 4, 6	(2, 6, 10)
2	1, 2, 3, 10, 8, 7	(2, 9, 16)
3	9, 11	(6, 9, 12)
4	12	(52, 54, 56)
Line Efficiency = (0.708955, 0.789063, 0.877049)		
Idle Percentage = (21.83099, 27.85714, 34.05797)		

Table 4. Results from Fuzzy COMSOAL – Test Problem 1

When comparing the two solutions based on the performance criteria of line efficiency and idle percentage, the most likely value for the fuzzy values of these two metrics was used. The most likely value for the fuzzy line efficiency for the fuzzy COMSOAL method is 0.789063 or 78.9%. This is the same value that was calculated for the line efficiency of the original method. The most likely value for the fuzzy line percentage was determined to be 27.85714 percent. This directly coincides with the line percentage value calculated in Askin and Standridge [2].

The second test problem used for validation of the new fuzzy method was taken from the study by Lee et. al [11]. Again, the original problem contained crisp time values, so the variables were again transformed to be represented as triangular fuzzy numbers. The solution generated in the original study is shown in Table 5, along with the values for the line efficiency and idle percentage calculations. The results obtained using the new fuzzy COMSOAL method are shown in Table 6. The fuzzy line efficiency and idle percentage values are also given.

The fuzzy COMSOAL generated a slightly different solution for task assignments when compared to the original method. However, the values for the line efficiencies and idle percentages are the same.

The third example problem used to validate the fuzzy COMSOAL procedure was taken from a study by Tsujimura et. al [15]. Unlike the previous two examples, the processing times were already represented by triangular fuzzy numbers.

Workstation	Assigned Task	Idle Time
1	1, 4, 3, 6	1
2	8, 11, 12, 15, 14	0
3	2, 5, 7	2
4	9, 10, 13, 16	3
Line Efficiency = 0.931818		
Idle Percentage = 6.818182		

Table 5. Results from Original Crisp Solution Method – Test Problem 2

Workstation	Assigned Task	Fuzzy Idle Time
1	1, 3, 4, 2	(0, 0, 5)
2	5, 6, 7	(0, 3, 7)
3	8, 9, 11, 10	(0, 3, 8)
4	14, 12, 13, 15, 16	(0, 0, 6)
Line Efficiency = (0.634615, 0.931818, 1)		
Idle Percentage = (0, 6.818182, 30.95238)		

Table 6. Results from Fuzzy COMSOAL – Test Problem 2

Workstation	Assigned Task	Fuzzy Idle Time
1	1, 2, 4, 6	(4, 11, 19)
2	5, 3, 8, 10	(9, 15, 25)
3	7, 11, 9	(12, 17, 22)
4	12	(30, 32, 35)
Line Efficiency = (0.57, 0.80, 1)		
Idle Percentage = (26.9608, 37.5, 51.53061)		

Table 7. Results from Original Solution Method – Test Problem 3

However, the cycle time was originally crisp, so, it had to be adapted to the fuzzy domain. The task assignments and performance criteria generated in the Tsujimura et. al's [15] study are shown in Table 7. The results from solving this example problem using the new fuzzy COMSOAL method are shown in Table 8.

Workstation	Assigned Task	Fuzzy Idle Time
1	1, 5, 4, 2, 7	(0, 6, 15)
2	6, 3, 9, 8	(6, 12, 20)
3	10, 11, 12	(2, 7, 15)
Line Efficiency = (0.647799, 0.94697, 1)		
Idle Percentage = (5.228758, 16.6667, 34.01361)		

Table 8. Results from Fuzzy COMSOAL – Test Problem 3

The result tables show two very different solutions for the test data set. The task assignments are different for the two models, and the fuzzy procedure produced one less workstation when compared to the original model. This point alone shows that the fuzzy COMSOAL has generated a more efficient solution than the previous method. Since the original solution was contained in the fuzzy domain, a direct comparison of the performance metrics can be performed. When examining the values for the two models' line efficiencies, it is clear that the fuzzy COMSOAL procedure has generated a more efficient line balance. The calculations for the idle percentages are also lower in the new method. This shows that there is less idle time contained at each workstation when the problem is solved with the fuzzy COMSOAL.

5.3 Validation of the Fuzzy RPWT

The validation phase for the fuzzy RPWT heuristic was performed in a similar manner to that of the COMSOAL method. The same three example problems were solved using the new RPWT technique.

As mentioned earlier, the first test problem was drawn from Askin and Standridge [2]. The time variables were transformed so that they could be represented as triangular fuzzy numbers. The results from the original solution method can be seen in Table 3 (same as the one generated by COMSOAL). The results from solving the test problem through the new RPWT method are shown in Table 9.

Workstation	Assigned Task	Fuzzy Idle Time
1	1, 4, 2, 3, 7	(0, 3, 9)
2	6, 8, 5, 9	(0, 2, 7)
3	11, 10, 12	(0, 3, 7)
Line Efficiency = (0.87963, 0.990196, 1)		
Idle Percentage = (0, 3.809524, 11.1111)		

Table 9. Results from Fuzzy RPWT – Test Problem 1

It seems that the two methods arrived at different solutions to the problem. The fuzzy RPWT method generated a solution that requires one less workstation than that of the original method. The most likely value for the fuzzy line efficiency is greater than the crisp efficiency in the original solution. Also, the idle percentage for the fuzzy method is smaller than that of the current line balance.

The second example problem was drawn from the study by Lee et. al [11]. The processing and cycle times were transformed into the same fuzzy numbers as in the validation of the COMSOAL technique. The original solution for this problem can be seen in Table 5. When the problem was solved using the fuzzy RPWT, the obtained results were as shown in Table 10.

Workstation	Assigned Task	Fuzzy Idle Time
1	1, 2, 4	(0, 2, 6)
2	5, 7, 3, 9	(0, 0, 5)
3	6, 8,10, 11	(0, 4, 9)
4	13, 12, 14, 15, 16	(0, 0, 6)
Line Efficiency = (0.634615, 0.931818, 1)		
Idle Percentage = (0, 6.818182, 30.9524)		

Table 10 Results from Fuzzy RPWT – Test Problem 2

By comparing the most likely value of the fuzzy idle percentage to that of the original solution, it is seen that these values are also identical. This shows that the fuzzy RPWT has produced a solution that is equivalent to that arrived at by the original study, with respect to the selected performance metrics.

The last test data set used to validate the fuzzy RPWT method also came from the work by Tsujimura et. al [15]. The solution for this problem from the original study can be seen in Table 7. When the problem is solved by the fuzzy RPWT method, the results obtained were as follows (Table 11):

Workstation	Assigned Task	Fuzzy Idle Time
1	1, 4, 2, 3, 5, 8	(1, 9, 20)
2	7, 10, 6	(0, 1, 9)
3	9, 11, 12	(11, 15, 21)
Line Efficiency = (0.624242, 0.85034, 1)		
Idle Percentage = (7.843137, 16.6667, 34.0136)		

Table 11. Results from Fuzzy RPWT – Test Problem 3

The fuzzy RPWT method requires only 3 workstations while the original crisp technique called for 4. The fuzzy line efficiency is higher in the RPWT, while the fuzzy idle percentage is lower.

After testing the new fuzzy COMSOAL and RPWT methods with data sets from the available literature, it was shown that these heuristics are very capable of solving the assembly line balancing problem. In all cases, the fuzzy methods generated solutions that were equally as efficient as the results drawn from the original procedures.

5. Conclusions

Two existing heuristics were successfully modified to model the variability and ambiguity imbedded in the stochastic ALBP. Time variables were represented by triangular fuzzy numbers, which allows the practitioner to account for the ambiguity in assigning processing and cycle times, while still maintaining the variability of the stochastic environment. The COMSOAL and Ranked Positional Weighting Technique solution methods were then transformed to solve the ALBP with fuzzy operating times. The fuzzy heuristics were then automated via Visual Basic. Three test example problems from the available literature were used to successfully validate the constructed fuzzy techniques. Thus, a viable alternative approach to solving the stochastic assembly line balancing problem was developed.

References

[1] Arcus, A.L., COMSOAL: A computer method of sequencing operations for assembly lines, International Journal of Production Research, 4 (1966) 25-32.

[2] Askin, R., and Standridge, C., Modeling and Analysis of Manufacturing Systems, John Wiley & Sons, Inc., New York, 1993.

[3] Baybars, Ilker, A survey of exact algorithms for the simple assembly line balancing problem, Management Science, 32 (1986) 11-17.

[4] Chase, R.B., Survey of paced assembly systems, Industrial Engineering, 6 (1974) 82-90.

[5] Erel, Erdal, and Sarin, S.C., A survey of the assembly line balancing procedures, Production Planning and Control, 9 (1998) 34-42.

[6] Gen, M., Tsujimura, Y., and Li, Y., Fuzzy Assembly Line Balancing Using Genetic Algorithms, Computers and Industrial Engineering, 31 (1996) 49-52.

[7] Gutjahr, A.L., and Nemhauser, G.L., An algorithm for the balancing problem, Management Science, 11 (1964) 23-35.

[8] Helgeson, W.B., and Birnie, D.P., Assembly line balancing using the ranked positional weighting technique, Journal of Industrial Engineering, 12 (1961) 18-27.

[9] Hong, T., and Chuang, T., A new triangular fuzzy Johnson algorithm, *Computers & Industrial Engineering*, **36** (1999) 75-79.

[10] Kao, E.P.C., A preference order dynamic program for stochastic assembly line balancing, *Management Science*, **22** (1976) 19-24.

[11] Lee, T.O., Kim, Y., and Kim, Y.K., Two-sided assembly line balancing to maximize work relatedness and slackness, *Computers & Industrial Engineering*, **40** (2001) 38-41.

[12] Milas, G., Assembly line balancing: Let's remove the mystery, *Industrial Engineering*, **22** (1990) 12-18.

[13] Salveson, M.E., The assembly line balancing problem, *Journal of Industrial Engineering*, **6** (1955) 62-69.

[14] Sury, R. J., Aspects of assembly line balancing, *International Journal of Production Research*, **9** (1971) 8-14.

[15] Tsujimura, Y., Gen, M., and Kubota, E., Solving Fuzzy Assembly-line Balancing Problem with Genetic Algorithms, *Computers and Industrial Engineering*, **29** (1995) 62-69.