TJFS: Turkish Journal of Fuzzy Systems (eISSN: 1309–1190)
An Official Journal of Turkish Fuzzy Systems Association
Vol.4, No.2, pp. 48-67, 2013

A Hybrid Approach for Fuzzy Clustering and Assembly Line Balancing Problems

Rifat Gürcan Özdemir*
İstanbul Kültür University, Faculty of Engineering,
Department of Industrial Engineering, 34156 İstanbul, Turkey
E-mail: rg.ozdemir@iku.edu.tr, *Corresponding author

Murat Demirer
İstanbul Gelişim University, Faculty of Engineering,
Department of Computer Engineering, 34315 İstanbul, Turkey
E-mail: m.demirer@iku.edu.tr

Mehmet Yahya Durak
İstanbul Kültür University, Faculty of Engineering,
Department of Industrial Engineering, 34156 İstanbul, Turkey
E-mail: y.durak@iku.edu.tr

Filiz Fakı
İstanbul Kültür University, Faculty of Engineering,
Department of Industrial Engineering, 34156 İstanbul, Turkey
E-mail: f.avci@iku.edu.tr

Ethem Tarhan
İstanbul Kültür University, Faculty of Engineering,
Department of Industrial Engineering, 34156 İstanbul, Turkey
E-mail: e.tarhan@iku.edu.tr

Abstract

Assembly lines are recently approaching a new model of flow-line production system as low volume production of customized products called mass customization. This study addresses the problems of component clustering under sub-assemblies and line balancing, simultaneously. Fuzzy C-Means clustering algorithm will help to find clusters of components which configure sub-assemblies. The clustering algorithm results are then incorporated with a mathematical model which solves assembly line balancing optimally. The entire approach is then implemented in automotive industry with given example.

Keywords: Fuzzy logic, Fuzzy clustering, Assembly line balancing.

1. Introduction

The problems of component clustering under sub-assemblies and line balancing are solved in a two stage approach. The component clustering is introduced as an unsupervised learning method based on fuzzy c-means which was introduced by Bezdek (1974). This widely used method here refers to the assignment of automotive components into clusters (sub-assemblies) so the components belonging to the same group are similar to each other for a given threshold which specifies the required common attributes of the components in the same cluster. Generating sub-assemblies provides to minimize the balance delay and minimize the number of stations like conveyor belt or a similar material handling equipment. The first published paper of the assembly line balancing problem was made by Salveson (1955) who suggested a linear programming solution. In a pure linear programming approach ALB problem falls into the NP hard class of combinatorial optimization problem (Gutjahr & Nemhauser, 1964). For that reason, efficient hybrid algorithms which make use of benefits of two algorithms has been developed for obtaining optimal solutions. The balancing problems are considered in the literature and some classification schemes are given (Ghosh & Gagnon, 1989; Scholl & Becker, 2003). The most common assembly line balancing problem is single line assembly line balancing problem (Baybars, 1986; Boysen et al., 2007). In single model assembly line, assemblers work on the identical design of the same product. One model of product is produced in one line. However, in recent years, most of the studies address mixed model assembly line balancing problems in which more than one models are produced interchangeably on the same line. To the best of our knowledge, there is no study in the literature addressing component clustering under sub-assemblies and assembly line balancing problems simultaneously. In this study, we consider both problems in a two stage approach in which a fuzzy clustering method is used for determining sub-assemblies and the results are input to a mathematical model in the second stage in order to balance assembly line optimally.

In subsequent sections of the paper, the problem definition is presented in Section 2, the binary integer mathematical formulation of the problem is given in Section 3, fuzzy c-means algorithm for component clustering is presented in Section 4, Implementation of the proposed approach is given for an industrial problem in Section 5, and finally the study is concluded in Section 6.

2. Problem Definition

Sub-assembly is a group of parts that is combined with base part in order to form of a finished assembly. Generating sub-assemblies will provide cycle time minimization due to savings on task times to be performed in the line. In assembly line, raw materials (components) and semi-finished products (sub-assemblies) are combined together and assembled on base product. Total time and costs of the finished product may differ due to sub-assembly selection. In literature, assembly line balancing and sub-assembly selection problems are addressed separately; however, optimal solution to one affects another. Therefore, this study considers both problems as one and proposes an approach solving jointly. Before sub-assembly selection, components should be grouped with respect to some manufacturing and technological criteria which are vague to be processed clearly. The sub-assembly alternatives are generated based on the clustering

results of components. The generated sub-assemblies and other assembly line data are input to the mathematical model for solving the above mentioned problems optimally. The clustering problem for generating sub-assembly alternatives has a fuzzy nature, so the paper proposes a fuzzy approach for the first step of the study. Clustering partitions $\mathcal{R}^3$dimensional attributed data space into "homogenous" clusters. Then, each 3-dimensional vector (Station No, Assembly Position, and Compatible Component) is more similar into each other in Euclidian sense within the same cluster (group). The goal is to find substructures of the components in the data which contains 90 components. If we were using Crisp clustering method, each component vector would belong to one and only one cluster. However, it is difficult to find distinct and clear boundary between clusters. For example, compatible component {7} which is a subset ($\subset$) of {1,2,3,6,7} is used in the station number {2} and assembly position {1}, while same compatible component can be used in another station {6} and assembly position {5}. Component {2,90} are with same compatible component {7}. Instead, we implemented fuzzy clustering each component vector may partially belong to more than one cluster, and the membership degrees/grades lie in between [0,1]. The resulting sub-assemblies and line balancing data are processed in a mathematical model which will be presented in the following section to obtain optimal solution to the problem addressed here.

3. Binary Integer Linear Mathematical Formulation of the Problem

In this part, assumptions, notation of the problem (indices, parameters, variables) and binary integer programming formulation is indicated.

Assumptions are as flows:

- Number of workers in each station is constant.
- The required times to perform tasks and sub-assemblies are known with certainty.
- Cycle time cannot be changed during the period.
- Tasks are performed in their assigned stations.
- The notation of the mathematical model is mentioned below;

Indices

i= Task index (i=1,..,N)
j= Station index (j=1,…,J)
k= Sub-assembly index (k=1,..,Km)
m= Model index

Parameters
t_{im}= Assembly duration for performing task i at model m
ts_k= Assembly duration of sub-assembly k
c= Cycle time
q_m= Quantity of model m to be produced in the period
cs_{km}= Production cost of sub-assembly k ($/unit) of model m
w= Wage of assembly operator employed in the line ($/min)

f= Fixed installation cost of a station in the line
B= Big number
S_m= Set of sub-assemblies produced for model m
ST_k= Set of tasks required to be performed to make sub-assembly k
$SPTT_i$= Set of tasks immediately precedes task i
$SPTS_i$= Set of sub-assemblies immediately precedes task i
$SPSS_k$= Set of sub-assemblies immediately precedes sub-assembly k
$SPST_k$= Set of tasks immediately precedes sub-assembly k

Variables

And the following binary integer variables are defined for every station j and for every task i:

$$X_{ij} = \begin{cases} 1, & \text{if task } i \text{ is assigned to station } j \\ 0, & \text{otherwise} \end{cases}$$

$$Y_{kj} = \begin{cases} 1, & \text{if sub-assembly } k \text{ is assigned to station } j \\ 0, & \text{otherwise} \end{cases}$$

$$R_j = \begin{cases} 1, & \text{if station } j \text{ is installed} \\ 0, & \text{otherwise} \end{cases}$$

$$XA_i = \begin{cases} 1, & \text{if task } i \text{ is assigned to line} \\ 0, & \text{otherwise} \end{cases}$$

$$YA_k = \begin{cases} 1, & \text{if sub-assemly } k \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

Objective Function

$$Min\ Z = w \cdot \sum_{j=1}^{J} \sum_{i=1}^{N} \sum_{j=1}^{M} (q_m t_{im} X_{ij}) + \sum_{j=1}^{J} \sum_{m=1}^{M} \sum_{k=1}^{K} (q_m cs_{km} Y_{kj}) + f \cdot \sum_{j=1}^{J} R_j \quad (1)$$

Subject To

$$\sum_{j=1}^{J} X_{ij} = 1,\ \forall i\ \notin \{\bigcup_{k=1}^{K} ST_k\} \quad (2)$$

$$\sum_{j=1}^{J} X_{ij} = 1 - \sum_{j=1}^{J} Y_{kj},\ \forall k\ and\ i \in ST_k \quad (3)$$

$$\sum_{j=1}^{J} Y_{kj} \leq 1, \forall k \quad (4)$$

$$\sum_{j=1}^{J} j.X_{hj} \leq \sum_{j=1}^{J} j.X_{ij} + B(1 - XA_i), for\ h \in SPTT_i \quad (5)$$

$$\sum_{j=1}^{J} j.Y_{kj} \leq \sum_{j=1}^{J} j.X_{ij} + B(1 - XA_i), for\ k \in SPTS_i \quad (6)$$

$$\sum_{j=1}^{J} j.Y_{gi} \leq \sum_{j=1}^{J} j.YA_k + B(1 - YA_k), for\ g \in SPSS_k \quad (7)$$

$$\sum_{j=1}^{J} j.X_{ij} \leq \sum_{j=1}^{J} j.Y_{kj} + B(1 - YA_k), \forall k\ and\ i \in SPST_k \quad (8)$$

$$\sum_{i=1}^{i} t_{im}X_{ij} + \sum_{k=1}^{K} ts_k Y_{kj} \leq c.R_j,\ \forall j \quad (9)$$

$$\sum_{j=1}^{J} X_{ij} = XA_i\ , \forall i \quad (10)$$

$$\sum_{j=1}^{J} Y_{kj} = YA_k\ , \forall k \quad (11)$$

$$R_j \geq R_{j+1} \quad , j = 1, \ldots, j-1 \quad (12)$$

$$X_{ij}, Y_{ij}, XA_i, YA_k, R_j = \{0,1\} \quad , \forall i, k, j \quad (13)$$

The equation (1) shows the objective function of our formulation which involves three parts as total costs for incurred due to sub-assemblies, total in-house production cost and total fixed cost for installation of all stations in the line. Constraint (2) ensures that each task which is not part of any sub-assembly should be assigned to a station. This means that if a task is a part of any sub-assembly, it can be assigned to a sub-assembly area. Constraint set (3) implies that each task of an unassigned sub-assembly should be assigned to a station. Constraint set (4) represents that each sub- assembly can be supplied to only one station. Constraint sets (5 - 8) show the precedence relations among tasks, between tasks and sub-assemblies and among sub-assemblies, respectively. For example, according to equation (5), if task *h* has a precedence relationship with task *i*, this means task *h* should be processed before task *i*. This is the reason why task *i* cannot be assigned before task *h*. In equation (6) if there is a precedence relationship between sub-assembly *k* and task *i*, it is shown that sub-assembly *k* should be assigned before task *i*. The equation (7) represents that if a precedence relationship exists between sub-assembly *g* and sub- assembly *k*; before assigning sub-assembly *k*, sub-assembly *g* cannot be assigned. In equation (8) according to precedence relations, the task *i* should be processed before sub- assembly *k*. Constraint (9) ensures that station content cannot exceed the predetermined cycle time. It is shown in the equation (10) that if task *i* is assigned to a station in the line, task assignment variable *XA* takes value of 1, otherwise 0. Constraint set (11) implies that if task *k* is selected, sub-assembly selection variable *YA* takes value of 1 and otherwise 0. According to constraint (12) it is shown in the equation that stations are opened sequentially. Finally constraint set (13) defines the decision variables. According to constraint (13) it is specified the binary decision variables. This means that the mentioned variables can be 0 or 1.

4. Fuzzy C-Means Algorithm

In FCM method, we will partition set of *n*=90 component vectors $o = \{\vec{o}_1, \vec{o}_2 \ldots \vec{o}_{90}\}$ in $\Re^3$dimensional attribute space (Station No, Assembly Position, Compatible Component) into *s* (1<*s*<*n*) fuzzy clusters with $Z = \{\vec{z}_1, \vec{z}_2, .., \vec{z}_s\}$ where *s*=5 cluster

centers. The fuzzy clustering of components is described by membership matrix μ with n=90 rows and s=5 columns. Here, n is the number of components and s is the number of clusters. μ_{ij}, the membership value in the i^{th} row and j^{th} column. Each value denotes the degree of association or membership function of the *i.* component with the j. Sub-Assembly cluster. The properties of membership function matrix μ are subject to the constraints as follows.

$$\mu_{ij}\epsilon[0,1], \forall i = 1,2, \dots n = 90;\ \forall j = 1,2, \dots s = 5$$

$$\sum_{j=1}^{s} \mu_{ij} = 1, \forall i = 1,2, \dots n = 90;$$

$$0 < \sum_{i=1}^{n} \mu_{ij} < n, \forall j = 1,2., \dots s = 5$$

The objective function of Fuzzy C-Means Clustering is to minimize of the following equation which is defined as objective function.

$J_e = \sum_{j=1}^{s} \sum_{i=1}^{n} \mu_{ij}^{e}\, d_{ij}$ where distance $d_{ij} = \left\|\vec{o}_i - \vec{z}_j\right\|$. In this equation, $e(e > 1)$ is a scalar real value which represents weighting exponent. It controls the fuzziness of the estimated clusters. d_{ij} is the Euclidian Distance from automobile components $o_j$to the cluster centers z_j as both in $\Re^3$ dimensional vectors. Each element of the vectors (o_i, z_j) represent Station No, Assembly Position, Compatible Component. The z_j centroid of the j^{th} cluster is obtained using the following equation.

$$\vec{z}_j = \frac{\sum_{i=1}^{n} \vec{\mu}_{ij}^{e}\, \vec{o}_i}{\sum_{i=1}^{n} \vec{\mu}_{ij}^{e}}$$

Fuzzy C-Means Algorithm (Bezdek, 1974; Izakian & Abraham, 2011) is iterative and will be given by the following algorithm.

Algorithm

1. Select e(e>1) exponent value; initialize the membership function values,$i = 1,2, \dots n; j = 1,2, \dots s$. The correct value of e can be selected using a criterion which allows an optimal upper bound for the exponent, e. This is calculated a priori directly from the data matrix (Yu et al., 2004).
2. Compute the cluster centers $z_j, j = 1,2, \dots c$ according to above equation.
3. Compute Euclidian Distance d_{ij}, $i = 1,2, \dots n$ and $j = 1,2, \dots s$ which is optimal for recovering hyperspherical clusters. It is still suboptimal when the clusters covariance is high which depends on high correlation among attributes (features).
4. Update the membership function μ_{ij}, $i = 1,2, \dots n$ and $j = 1,2, \dots s$ according to the following equation

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{s}\left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{e-1}}}$$

5. The update formulae are derived by setting partial derivative of objective function equal to zero under constraints.
6. If the relative change of the centroid,μ_{ij} vector becomes small or objective function cannot be minimized more than prefixed accuracy ($\epsilon = 10^{-5}$) or when the number of iterations exceeds a maximum allowed value like 100 stops and go to step 2.

Several parameters influence Fuzzy C-Means outcome and performance: feature space dimensionality, true number of the clusters, clusters' overlap and fuzzifier exponent, e . Even Fuzzy C-Means algorithm is very popular clustering algorithm but it is sensitive to initial values and falls into local minima of the objective function

5. Implementation

This section presents a case study for implementation of the proposed approach. An assembly line of automotive manufacturer is considered and the pertinent data are given in the following tables. A sample set of components from the line are presented in Table 1. Full data for components with their precedence relations are given in appendix.

Table 1. Components and their attributes

COMPONENT NO	ATTRIBUTES		
	Station No	Assembly Position	Compatible Component
1	20	1	1,2,3,4,5
2	2	1	1,2,3,6,7
3	1	3	8,9
...	...	...	...
88	15	3	12,25,79,87
89	8	7	15,56,88
90	6	5	7,46,58,66,79,82,89

The Fuzzy C-Means clustering algorithm is then applied to the given components with the three attributes listed above and the results are obtained as given in Table 2.

Table 2. Sub-assemblies and components assigned

Sub-Assembly	Component
1	1,2,3,4
2	5,9,12,13,46
3	8,37,67,73
4	34,62,64,56,78,87
5	27,52,57,68,84

The results given in the above table shows that 24 components are grouped into 5 clusters which are sub-assemblies to be combined with the base product in the line.

These sub-assemblies given in the above table is then input to the mathematical model and assembly line is balanced as given in the Tables 3 and 4.

Table 3. Results of the mathematical problem between station 1-8.

Station	Assigned Task	Assigned SA	Station Load (sec.)	Labor Cost ($)	Fixed Cost ($)	SA Cost ($)	Total Cost ($)
1	3,4,8,9,20,23,49,55,68, 71,72,74,81,96,106,107 ,120,150,152,160,179,1 86,187,188,189,190,21 1,215,223,235,238,240	-	299	10354	5000	0	15354
2	12,18,56,57,82,92,110, 135,143,161,170,171,1 75,178	-	298	10048	5000	0	15048
3	10,13,62,84,141,142,14 4,173,192,209,210	SA1	257	7804	5000	840	13644
4	24,26,46,77,91,111,112 ,116,132,133,159,180	-	298	9647	5000	0	14647
5	34,35,43,51,87,118,131 ,174,191,213,222	-	248	7254	5000	0	12254
6	6,14,17,85,121,134,162 ,163,164,165,166,167,1 69	-	276	10200	5000	0	15200
7	19,25,40,52,80,83,90,1 54,155,224,232	-	283	9896	5000	0	14896
8	16,76,146,147,148,168, 176,216,236,237	-	297	10870	5000	0	15870

As seen table 3 and 4, three sub-assemblies are selected and inserted into the line at station 3, 13 and 16 respectively. The table also provides cost calculation for labor, generation of sub-assemblies, and setup of station.

Table 4. Result of the mathematical problem between station 9-16.

Station	Assigned Task	Assigned SA	Station Load (sec.)	Labor Cost ($)	Fixed Cost ($)	SA Cost ($)	Total Cost ($)
9	15,30,36,38,44,63,122,126,127,128,129,130,151,194,206	-	283	7390	5000	0	12390
10	45,64,66,69,89,136,207,217,242	-	295	11525	5000	0	16525
11	1,2,5,22,37,47,75,137,138,153,177,185,196,234,239	-	299	9728	5000	0	14728
12	27,31,48,53,73,93,94,113,156,212,218,219	-	281	10360	5000	0	15360
13	7,11,21,33,41,54,65,67,95,98,114,139,183,198,221,233	SA3	219	8235	5000	2430	15665
14	39,42,50,70,78,79,86,99,119,184,195,199,201,225,241	-	219	10850	5000	0	15850
“	28,115,123,124,125,145,149,157,202,204,205,208,220,228,229	-	228	8190	5000	0	13190
16	29,97,104,105,108,140,158,172,193,197,214,231,243,244	SA4	210	8275	5000	2670	15945

5. Conclusion

This study has developed a hybrid approach for the joint problems of component clustering and line balancing. The solution has been handled in two different stages as follows: first stage obtains the component clusters using fuzzy C-Means algorithm and the results are the input of the second stage. In the second stage a mathematical model is used to find assembly line balance optimally with consideration of sub-assembly selection. The model has been implemented and the results show that the proposed approach is promising to be used in real life problems.

References

Baybars, I., 1986. A survey of exact algorithms for the simple assembly line balancing problem, Management Science 32, 909–932.

Bezdek, J., 1974. Fuzzy Mathematics in pattern classification Ph.D thesis Ithaca, NY Cornell University.

Boysen, N., Fliedner, M., Scholl, A., 2007. A classification of assembly line balancing problems, European Journal of Operational Research 183, 674–693.

Ghosh S. and R. J. Gagnon., 1989. A Comprehensive Literature Review and Analysis of the Design, Balancing and Scheduling of Assembly Systems. International Journal of Production Research. Vol. 27, No.4: pp.637-670.

Gutjahr, A. L., Nemhauser, G.L., 1964. An algorithm for the line balancing problem. Management Science. Vol. 11, No.2: pp. 08-315.

Izakian, H., Abraham, A., 2011. Fuzzy C-means and fuzzy swarm for fuzzy clustering problem, Expert Systems and Applications 38, 1835-1838.

Salveson, M.E.,1955. The Assembly Line Balancing Problem. The Journal of Industrial Engineering.Vol. 6,No.3: pp.18-25.

Scholl, A. and Becker, C., 2003. A note on an exact method for cost-oriented assembly line balancing. Jenar Schriften zur Wirtschaftswissenschaft 22/2003, University of Jena.

Yu, J., Cheng, Q., Huang, H., 2004. Analysis of the weighting exponent in the FCM, IEEE Transaction on Systems, Man and Cybernetics –Part B:Cybernetics 34 (1), 634-639.

APPENDIX

Table 5. The durations, codes and precedence relations of tasks 1-36.

Task No.	Task Code	Station No.	Task Time	Preceded by
1	FBC 07400	1	40	
2	PROCESS	1	8	1
3	FBL 00015	1	30	
4	FBL 00020	1	10	3
5	FTZ 05002	1	7	2
6	FBC 07400	1	40	
7	PROCESS	1	8	6
8	FBL 00015	1	10	
9	FBL 00020	1	10	8
10	FML 00100	1	19	9
11	FTZ 05002	1	8	7,9
12	FPA 00001	1	9	
13	FBC 01100	2	15	4
14	FBC 03545	2	120	4
15	FBC 04050	2	60	4
16	FBC 04310	2	94	4
17	FBC 04320	2	96	4
18	FML 04235	2	5	4
19	FBF 06700	2	8	4
20	FTZ 05000	2	16	4
21	FBL 00210	2	12	24
22	FBF 07000	2	25	4
23	FBS 01520	2	4	4
24	FTF 02071	2	24	4
25	FWB 01900	2	40	9
26	FWB 01905	2	20	4
27	FWB 01910	2	20	9
28	FBS 01570	2	4	9
29	FBL 00210	2	12	31
30	FBS 01520	2	4	9
31	FTF 02071	2	24	9
32	FER 08700	2	25	9
33	FBS 01570	2	4	9
34	FER 08750	2	32	9
35	FAH 06250	3	5	9
36	FBS 00050	3	4	4

Table 6. The durations, codes and precedence relations of tasks 37-74

Task No.	Task Code	Station No.	Task Time	Preceded by
37	FNU 00350	3	28	4
38	FTN 00200	3	23	4
39	FSO 00206	3	5	26
40	FTN 00210	3	8	4
41	FNA 01450	3	8	4
42	FTN 00140	3	15	39
43	FTP 00020	3	22	4
44	FAH 06250	3	6	9
45	FBL 00479	3	4	9
46	FTN 00200	3	23	9
47	FML 00500	3	7	10
48	FSO 00206	3	5	27
49	FTN 00210	3	8	9
50	FTN 00140	3	15	48
51	FTP 00020	3	22	9
52	FBS 03705	4	30	4
53	FML 02000	4	8	9
54	FSW 01200	4	28	4
55	FBC 08500	4	47	4
56	FSW 01220	4	28	55
57	FWB 03500	4	15	56
58	FBC 09001	4	7	57
59	FBC 09720	4	10	57
60	FBC 00005	4	10	59
61	FBC 09725	4	10	60
62	FWB 01925	4	7	61
63	FWB 03550	4	18	62
64	FWB 03560	4	13	63
65	FBS 03705	4	20	9
66	FML 02500	4	65	64
67	FML 02800	4	20	65
68	FML 03510	4	35	9
69	FBS 03705	4	20	66,68
70	FBS 03705	4		69
71	FWB 03100	4	7	9
72	FSO 00250	5	12	9
73	FML 03510	5	35	4
74	FSW 01300	5	78	4

Table 7. The durations, codes and precedence relations of tasks 75-112

Task No.	Task Code	Station No.	Task Time	Preceded by
75	FTF 02079	5	27	4
76	FTN 00280	5	15	4
77	FRC 01850	5	15	18
78	FTN 00270	5	36	4
79	FBC 09900	6	5	4
80	FML 00150	6	5	9
81	FML 04245	6	51	9
82	FML 04250	6	49	81
83	FML 04250	6	22	82
84	FWL 00100	6		82
85	FSW 01320	6	20	9
86	FEF 04060	6	45	84
87	FBC 09900	7	5	9
88	FBC 07500	7	60	86
89	FSW 01320	7	14	69
90	PROCESS	7	32	84
91	FXL 01450	7	25	4
92	FML 03520	7	40	4
93	FML 03530	7	5	73
94	FML 03540	7	14	93
95	FBC 09800	7	3	94
96	FML 03500	7	3	4
97	FBC 09500	7	15	95
98	FSW 01221	7	45	89
99	FBS 00101	7	3	98
100	FML 00600	7		98
101	FML 02000	7	32	10
102	FML 02600	7	9	89
103	FML 02900	7	17	102
104	FML 03520	8	8	103
105	FML 03530	8	16	104
106	FML 03540	8	7	71
107	PROCESS	8	24	106
108	FBC 08800	8	33	107
109	FBC 08900	8	21	104
110	FBC 09600	8	20	109
111	FML 02900	8	20	110
112	FML 03520	8	40	111

Table 8. The durations, codes and precedence relations of tasks 112-150

Task No.	Task Code	Station No.	Task Time	Preceded by
113	FML 03530	8	5	93
114	FML 03540	8	14	113
115	FBC 09700	8	45	114
116	FBC 09785	8	5	112
117	FML 00080	8	10	9
118	FSD 01030	8	18	9
119	FEF 04060	8	40	116
120	FBC 08550	8	15	9
121	FML 03520	8	8	9
122	FML 03530	8	16	116
123	FML 03540	8	7	93
124	FBC 09705	8	17	123
125	FBC 09770	8	6	124
126	FBC 09000	8	19	122
127	FBC 09010	8	19	126
128	FBC 09600	8	17	127
129	FBC 09700	8	27	128
130	FSO 00207	9	14	129
131	FML 04251	9		112
132	FBC 00001	9	43	4
133	FML 04055	9	5	4
134	FLR 08850	9	10	9
135	FXL 01000	9	20	9
136	FSW 01305	9	43	130
137	FXL 01350	9	10	130
138	FXL 01200	9	15	77
139	FEG 07825	9	14	138
140	FML 04000	9	15	139
141	FML 04200	9	20	84
142	FML 04215	9	15	9
143	FML 04260	9	22	9
144	FWD 02300	9		4
145	FML 04210	9	16	141
146	FML 04220	9	21	9
147	FML 04270	9	4	4
148	FWL 01000	9	88	9
149	FTB 01110	9	55	145
150	FSW 03510	10	90	4

Table 9. The durations, codes and precedence relations of tasks 151-188

Task No.	Task Code	Station No.	Task Time	Preceded by
151	FBS 00300	10	45	4
152	FSP 00570	10	10	4
153	FSW 01100	10	60	9
154	FMW 07315	10	58	9
155	FFE 00030	10	15	9
156	FFE 00031	10	50	130
157	FTB 01080	10	73	156
158	FTB 01100	10	76	157
159	FTB 01120	10	48	9
160	FTS 09105	10	13	9
161	FTD 01019	10	10	143
162	FBC 09100	10	34	141
163	FBC 09200	10	15	162
164	FBC 09300	10	15	163
165	FTB 01110	10	55	164
166	FSW 01100	10	105	165
167	FBS 02080	10	26	166
168	FTB 01091	10	78	167
169	FSW 03520	10	30	85
170	FML 04050	10	26	4
171	FTB 01060	11	70	4
172	FFE 00250	11	15	39
173	FML 04230	11	57	152
174	FML 04240	11	26	9
175	FLR 08900	11	46	161
176	FXL 01450	11	60	168
177	FKR 03060	11	40	176
178	FBC 00060	11	35	175
179	FBS 01580	11	10	4
180	FBS 04039	11	12	4
181	FEF 04050	11	53	17
182	FWP 02200	11	18	180
183	FWB 07000	12	13	146
184	FWB 07000	12	13	9
185	FMW 07325	12	15	9
186	FWB 01250	12	17	4
187	FWB 01250	12	17	186
188	FBS 03905	12	15	187

Table 10. The durations, codes and precedence relations of tasks 189-226

Task No.	Task Code	Station No.	Task Time	Preceded by
189	PROCESS	12		188
190	FYF 03600	12	49	189
191	FTF 02000	12	15	9
192	FMW 09200	12	5	4
193	FTN 00130	12	13	192
194	FBS 00245	12	5	146
195	FBS 02490	12	5	4
196	FYF 03600	12	49	4
197	FTF 02000	12	15	4
198	FTN 00130	12	13	9
199	FBS 02490	12	5	198
200	FAH 04900	12		9
201	FBS 00240	13	5	9
202	FAH 05650	13		9
203	FCC 00100	13	28	28,33
204	FCC 02610	13	28	203
205	FJM 04010	13	30	204
206	FSW 01000	13	66	4
207	FWB 01000	13	8	4
208	FAB 05850	13	18	4
209	FAB 05800	13	16	9
210	FEF 04020	13	30	209
211	FTB 01070	14	82	9
212	FTB 01090	14	71	47
213	FBS 00710	14	10	9
214	FBS 00700	14	25	211
215	FKR 03060	14	20	211
216	FBL 00175	14	4	4
217	FBL 00180	14	4	216
218	FFE 00031	14	50	194
219	FBS 04400	14	20	169
220	FBS 04400	14	50	19
221	FLF 01107	14	50	22
222	FLF 01107	14	23	4
223	FMW 07300	15	63	9
224	FBC 04010	15	58	4,40
225	FBC 04020	15	52	49
226	FBF 00900	15	45	182

Table 11. The durations, codes and precedence relations of tasks 227-244

Task No.	Task Code	Station No.	Task Time	Preceded by
227	FML 01000	15	36	226
228	FNE 00100	15	25	205
229	FCC 02610	15	38	228
230	FWP 02520	15	23	229
231	FBU 03001	15	175	230
232	FBC 00070	15	32	9
233	FBU 03025	16	23	4
234	FSW 01000	16	66	185
235	FWB 01000	16	8	4
236	FWP 02200	16	8	132
237	FXL 01450	16	75	236
238	FGD 02040	17	70	
239	FGD 02050	17	60	236,238
240	FGD 02090	17	70	
241	FGD 02100	17	60	240
242	FGD 03040	17	70	9
243	FGD 03050	17	60	181
244	FZZ 05501	17	80	5…..243

Table 12. Full data for components 1-40

COMPONENT	ATTRIBUTES		
NO	Station No	Assembly Position	Compatible Component
1	20	1	1,2,3,4,5
2	2	1	1,2,3,6,7
3	1	3	8,9
4	17	5	1,2
5	16	3	56,59
6	3	2	25,45,56
7	5	2	90
8	4	4	33,42,82
9	7	6	11,16,46,54
10	9	5	12,13,37,46,77
11	2	1	9
12	8	2	10,88
13	5	4	10,63,64,65
14	15	8	19,24,35
15	12	5	89
16	13	5	9,23,30,74
17	4	4	38,41,52
18	18	1	21,28,49,84
19	2	4	14,51,62
20	19	6	69,80,87
21	15	5	18,36,40,48
22	5	3	35,50,53
23	4	1	16
24	12	2	14
25	13	4	6,88
26	17	6	60,61,75
27	8	5	39,60,78
28	4	2	18,70,85,87
29	15	1	33,35,54,64,73
30	12	5	16
31	4	6	68,80,86
32	5	5	61,71,77
33	7	1	8,29
34	15	4	42,53,65
35	2	2	14,22,29
36	13	6	21,43,45,59
37	14	2	10
38	18	2	17,81,83
39	12	1	27,48,57
40	11	5	21,68,76,85

Table 13. Full data for components 41-79

COMPONENT NO	ATTRIBUTES		
	Station No	**Assembly Position**	**Compatible Component**
41	3	5	17,55
42	5	4	8,34
43	4	5	36,55,56,67
44	20	3	63,72
45	4	2	6,36
46	13	6	9,10,90
47	4	2	79,86
48	7	5	21,39,78
49	17	2	18
50	4	1	22,52
51	5	1	19
52	18	4	17,50
53	19	7	22,34
54	11	6	9,29
55	16	6	6,41,43,83
56	4	5	5,43,89
57	5	2	39
58	2	4	90
59	11	4	5,36
60	6	5	26,27
61	14	6	26,32
62	13	1	19
63	2	6	13,44
64	16	7	13,29
65	4	6	13,34
66	10	7	90
67	2	4	43
68	11	4	31,40
69	18	2	20
70	20	2	28,81,84
71	9	3	32,76,83
72	19	6	44
73	4	6	29,82,84
74	5	1	16
75	6	3	26,86,87
76	17	5	40,71
77	13	6	10,32
78	2	5	27,48,88
79	4	3	47,90

Table 14. Full data for components 80-90

COMPONENT	ATTRIBUTES		
NO	Station No	Assembly Position	Compatible Component
80	12	2	20,31
82	15	6	8,73,90
83	13	1	38,55,71
84	17	1	18,70,73
85	4	4	28,40
86	7	4	31,47,75
87	20	4	20,28,75
88	15	3	12,25,79,89
89	8	7	15,56,88,90
90	6	5	7,46,58,66,79,82,89

TJFS: Turkish Journal of Fuzzy Systems (eISSN: 1309–1190)
An Official Journal of Turkish Fuzzy Systems Association
Vol.4, No.2, pp. 68-76, 2013

A Fuzzy Model of Software Project Effort Estimation

Oumout Chouseinoglou *
Hacettepe University, Faculty of Engineering,
Department of Industrial Engineering, 06800 Ankara, Turkey
E-mail: uhus@hacettepe.edu.tr
*Corresponding author

Özlem Müge Aydın
Hacettepe University, Faculty of Engineering,
Department of Industrial Engineering, 06800 Ankara, Turkey
E-mail: ozlemaydin@hacettepe.edu.tr

Abstract

Software project effort estimation is defined as one of the most difficult tasks in software engineering, embodying extensive uncertainty and vagueness in estimation parameters. In order to deal with this uncertainty, it has been suggested that the fuzzy approach can be employed in the effort estimation process. This work in progress paper is a proof of the concept that the estimation approach can be fuzzified. The proposed fuzzy estimation method is applied on a clustered ISBSG 11 dataset and it is concluded that the obtained results are acceptable.

Keywords: Software effort estimation, Fuzzy estimation, Software projects, COCOMO models

1. Introduction

Due to their special nature and complex processes, estimating the work-effort, cost and schedule of software development projects is one of the most difficult tasks in software project management. Software engineering, which can be defined as the application of a systematic, disciplined, and quantifiable approaches to the areas related to software, is trying to address these estimation difficulties. With that respect, several cost estimation models have been developed with the aim of constructively modeling the development processes and accurately predicting the cost of developing software. An exhaustive list of the different software effort and cost estimation studies is given by Jorgensen and Shepperd (2007).

Among the proposed effort estimation models, the most common ones are the algorithmic models, such as COCOMO and function points. As stated by Shepperd and Schofield (1997), the general structure of the model is in the form of

$$e = a_0 S^{a_1} \quad (1)$$

where e is effort, S is size, typically measured as lines of code (LOC) or estimated as function points (*fp*), α_0 is a productivity parameter and α_1 is an economies or dis-economies of scale. *fp* are a unit of measurement used in software engineering to express the amount of business functionality a software program is intended to provide to a user and they can be estimated prior to the development of the software program with the use of different counting methods, the most known being the COSMIC, NESMA and IFPUG.

Nevertheless, as stated by Xu and Khoshgoftaar (2004), no model has proven to be consistently successful in providing accurate effort and cost estimations, and despite all the efforts undertaken by the software engineering discipline, the estimation problems still exist today, resulting in delayed and over budget software. The main reason for this is that the inputs of any effort estimation formula, like the one given in Eq (1), are vague and are result of informed guessing rather than exact measurements (Musilek et al., 2000). Moreover, the information about the effort that is required to complete the software is often uncertain, imprecise or incomplete (Xu and Khoshgoftaar, 2004). Therefore, the vagueness and uncertainty in the effort estimation inputs necessitates the utilization of alternative approaches, such as fuzzy models. The sources of uncertainty in software cost and effort estimation models, and how a software project can be described as a fuzzy set are given by Musilek et al. (2000).

This paper is the initial part of a larger study aiming to develop a complete fuzzy model to estimate software project effort by utilizing fuzzy approach in all processes and parameters of the estimation. With respect to the overall aim, this study is a proof-of-concept, presenting that fuzzy approach even in a simple effort estimation model does not only generate acceptable results, but also provides the project management with a fuzzy number that can be used as the range of the expected effort, instead of a single value. The proposed approach is empirically validated in the International Software Benchmarking Standards Group (ISBSG) 11 dataset and the results are presented in detail.

The present paper is organized as follows: Section 2 presents a literature review regarding the use of fuzzy logic approaches in software effort and cost estimation models. Section 3 describes the fuzzy arithmetic and the model evaluation and quality measures that have been used in this paper. Section 4 details the fuzzy effort estimation proposed and lists the results obtained from the empirical evaluation. The last section presents conclusion and directions for future work.

2. Related work

There are several examples with respect to the use of fuzzy approaches and logic in software effort and cost estimation literature. Xu and Khoshgoftaar (2004) propose a fuzzy identification cost estimation model to deal with linguistic data, and automatically generate fuzzy membership functions and rules. By using the COCOMO81 project database, the authors cluster the project data with the use of fuzzy c-means and use as

inputs to the proposed model the cost driver attributes of the COCOMO model. By using a set of rules they have generated with the use of the Takagi-Sugeno models they calculate the values of the categories and subsequent membership functions they have devised. Finally they extract a crisp value from the fuzzy sets, similar to the defuzzification process, and they use the "Centroid of Areas" method to calculate the defuzziffication values. The authors investigate 63 project data from the COCOMO81 database and they empirically cluster them in 5 clusters. They conclude that the cost estimation accuracy of the fuzzy models generated by the proposed approach is significantly better than that of the COCOMO models. Musilek et al. (2000) propose the f-COCOMO model, a fuzzy set-based generalization of the COCOMO model. In f-COCOMO, instead of using a single number as the software size, the authors propose the use of a fuzzy number, which in return yields to another fuzzy number as the cost estimate. The authors conduct an analysis of their model with different membership functions such as the triangular and the parabolic fuzzy sets and they further implement the use of fuzzy numbers as parameters to the f-COCOMO. They experiment with 63 projects and propose that the fuzzy logic approach needs to be applied in other software cost and effort studies, such as the *fp*. Aroba et al. (2008) have proposed the use of fuzzy clustering for the development of segmented software cost estimation models, where a software project may belong to more than one segments. Their approach is tested on the ISBSG dataset, where they report their findings for clusters in the size of 11, 15 and 20.

Idri et al. (2001) propose the fuzzy analogy to be used to estimate the cost of software by providing a solution to the vagueness and impreciseness of the software attributes. Estimation by analogy is a four step process where first the similar cases are retrieved, the information gathered from them are reused, the proposed solution is revised and finally some of the parts of this experience are retained to be used in future projects. The authors use fuzzy logic and linguistic quantifiers in reasoning by analogy to estimate the effort of software projects and validate their approach by conducting an experiment on the COCOMO dataset. Similarly Azzeh et al. (2011) propose an analogy-based software effort estimation using fuzzy numbers, namely Generalized Fuzzy Number Software Estimation. They compute the similarity between two generalized fuzzy numbers based on their geometric distances, center of gravities and height of the generalized fuzzy numbers, and use fuzzy c-means to cluster the existing software project data. The estimations are conducted with the use of generalized fuzzy number operations and the effort of a project is estimated as a fuzzy number which is defuzzified with the method of center of gravity. The authors conduct empirical evaluations with the use of jack-knifing in benchmark software data from the ISBSG, Desharnais, Kemerer, Albrecht and COCOMO datasets. Azzeh et al. (2010) have also proposed one other estimation by analogy model that incorporates fuzzy set theory and grey relational analysis. In this model fuzzy set is employed to reduce uncertainty in distance measure between two tuples whereas the grey relational analysis is utilized as a problem solving method to assess the similarity between the tuples with a number of features. The authors compare their results with case-based reasoning, multiple linear regression and artificial neural networks, using the datasets given in their work (Azzeh et al. 2011).

Lopez-Martin et al. (2008) compare three personal fuzzy logic models to estimate the effort of small software programs, namely triangular, trapezoidal and Gaussian membership functions, with linear regression model. They develop the fuzzy logic and linear regression models using the data gathered from 105 small programs, and then the estimations generated by these models are compared with each other using 20 small programs.

3. Fuzzy arithmetic and model validation measures

For a triangular fuzzy number *M=(m, α, β)* let *m* be the mean value, and *α, β* be the left and right spreads, respectively. Membership function of *M* can be written as,

$$\mu_M(x) = \begin{cases} L(\frac{m-x}{\alpha}), x \le m \\ R(\frac{x-m}{\beta}), x \ge m \\ 0, otherwise \end{cases} \quad (2)$$

where *α, β > 0*. Bansal (2010) defines the multiplication of two fuzzy numbers *M=(m, α, β)* and *N=(n, γ, δ)* as,

$$M \otimes N \cong (mn, m\gamma + n\alpha - \alpha\gamma, m\delta + n\beta + \beta\delta) \quad (3)$$

and exponentiation function of two fuzzy numbers as,

$$M^N = (m^n, m^n - (m-\alpha)^{(n-\gamma)}, (m+\beta)^{(n+\delta)} - m^n) \quad (4)$$

The indicators to be used by software practitioners when comparing the results of different models are given by Kitchenham et al. (2001), and are used in a variety of studies investigated in this paper (Aroba et al. 2008, Azzeh et al. 2011, Azzeh et al. 2010, Idri et al. 2002).

Magnitude Relative Error (MRE) computes the absolute percentage of error between actual effort (*ea*) and estimated effort (*ee*), for each investigated project, as shown in Eq (5).

$$MRE_i = \frac{|ea_i - ee_i|}{ea_i} \quad (5)$$

On the other hand, Magnitude Error Relative (MER) is given in Eq (6).

$$MER_i = \frac{|ea_i - ee_i|}{ee_i} \quad (6)$$

Mean Magnitude Relative Error (MMRE) calculates the average of MRE over all investigated items (*n*), as shown in Eq (7). Similarly the Mean Magnitude of Error Relative (MMER) is given in Eq (8).

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{7}$$

$$MMER = \frac{1}{n}\sum_{i=1}^{n} MER_i \tag{8}$$

Finally, *PRED(q)* is used to count the percentage of estimates that fall within less than or equal to *q* of the actual values.

$$PRED(q) = \frac{\lambda}{N} \tag{9}$$

where λ is the number of projects where $MRE_i \leq q$ and *N* is the number of all estimates.

Aroba et al. (2008), referencing Conte et al. (1986), state that to evaluate the performance of a given model, a model whose $MMRE \leq 0.25$ and $PRED(0.25) \geq 0.75$ is considered to be a good one. In general, an estimation model with lower MMRE and higher PRED(*q*) can be interpreted that its derived estimates are more accurate than other models.

4. Fuzzy numbers for software project effort estimation

Crisp effort estimation function given in Eq (1) is calculated as a function of *fp*, assuming that *fp* value is exactly known. However, *fp* mostly depend on imprecise software attributes (Xu and Khoshgoftaar, 2004). In order to overcome this uncertainty, fuzzy logic is inserted into the model. As the main vagueness come from *fp* values, *fp* are defined by triangular fuzzy numbers and denoted as $\tilde{fp}$. Hence the estimated effort is achieved as a fuzzy number and denoted as $\tilde{e}$. Writing Eq (10) for fuzzy effort estimation, Eq (1) is rewritten as,

$$\tilde{e} = a_0 \, \tilde{fp}^{\,a_1} \tag{10}$$

In this study, in addition to *fp* values, a_o and a_1 parameters are also fuzzified due to their confidence intervals, which eventually give symmetric triangular fuzzy numbers. The final fuzzy effort estimation model is defined in the form of,

$$\tilde{e} = \tilde{a}_0 \, \tilde{fp}^{\,\tilde{a}_1} \tag{11}$$

5. Model adequacy checking

The described fuzzy methodology was applied over the ISBSG 11 dataset, to empirically validate its applicability. ISBSG dataset contains 5052 software projects of various types, with 118 attributes per project. As the ISBSG projects widely vary with respect to these attributes, in order to obtain a uniform dataset only the projects whose *fp* count is IFPUG and unadjusted *fp* rating is classified as A were selected, resulting to 2257 software projects. Moreover, as obtaining a single estimation function would not be appropriate for that final set of projects, prior to estimating the effort function, data was clustered according to both *fp* and effort values. SPSS 17 was used to cluster data set by *k*-means methodology. Number of clusters was empirically defined as 20 and 50 iterations were conducted at the initiation. Within the determined 20 clusters, the ones including at least 20 observations were taken into consideration (totally 2207 projects) and thus effort estimates depending on six clusters were calculated by non-linear regression model estimation in SPSS. Crisp parameter estimations a_o and a_1 are presented in the third and fifth columns of Table 1. As stated previously, due to the aforementioned uncertainty, *fp* values were considered as fuzzy numbers. In fuzzifying the *fp* in data set, the spreads were determined by extending the mean (crisp) *fp* by *±0,05 * fp*. Thus, crisp *fp* were converted to symmetric triangular fuzzy numbers by using the extension principle of fuzzy logic.

Table 1. Parameter estimations for clusters

Cluster	Number of observations	a_0		a_1	
		mean	spread	mean	spread
1	79	16180,877	3867,681	0,020	0,035
2	139	10276,013	1934,095	0,018	0,029
3	599	2508,763	319,284	0,026	0,023
4	1051	312,112	52,848	0,197	0,032
5	318	6448,994	1040,687	0,000	0,027
6	21	25349,009	6328,847	0,011	0,035

Spreads of each fuzzy number according to clusters are also given in Table 1.

For each cluster, both crisp and fuzzy effort estimations were calculated using each cluster's parameter estimates. Quality of the estimators were measured by MMRE, MMER and PRED(0,25) values and the results for crisp and fuzzy estimation models are given in Table 2.

At the last step of this study, six clusters were merged and accuracy errors, calculated by estimations according to their own estimators, were examined. Quality measures for overall data are presented in Table 3. It is seen in Table 3 that, fuzzy model is some worse than the crisp one, however, it is not such an inadequate model to be avoided.

Especially, in cases where lack of data or ill-defined data exists, fuzzy estimation model can be used to get satisfactory results.

Table 2. Estimation quality for crisp and fuzzy estimation models

Cluster	Measure	CrispModel	Fuzzy Model
1	MMRE	0,1058	0,1244
	MMER	0,1056	0,1116
	PRED(0,25)	1,0000	0,9494
2	MMRE	0,1308	0,1420
	MMER	0,1293	0,1311
	PRED(0,25)	0,9424	0,8633
3	MMRE	0,2405	0,2464
	MMER	0,2264	0,2245
	PRED(0,25)	0,5492	0,5525
4	MMRE	1,4277	1,4637
	MMER	0,4936	0,4833
	PRED(0,25)	0,2683	0,2712
5	MMRE	0,1771	0,1858
	MMER	0,1724	0,1718
	PRED(0,25)	0,7610	0,7673
6	MMRE	0,0500	0,0739
	MMER	0,0498	0,0667
	PRED(0,25)	1,0000	1,0000

Table 3. Estimation quality of overall data

Measure	Crisp Model	Fuzzy Model
MMRE	0,7832	0,8048
MMER	0,3337	0,3288
PRED(0,25)	0,4912	0,4875

Table 2 and 3 show that the calculated accuracy measures for both models are similar to each other. Although the proposed fuzzy model could not serve better results in all clusters, it should be kept in mind that fuzzy model achieved these similarities by using extended values. This extension eases decision makers in determining the *fp* and the other parameters. Moreover, within the 2207 examined software projects, 56,54% of the

actual efforts are taking place within the estimated fuzzy effort interval. With all these taken into account, the fuzzified model is concluded to be sufficient.

6. Conclusions and future work

Effort estimation is one of the most significant fields in software project management as it includes extensive uncertainty and vagueness, with various types of software projects, thus making generalizations impossible. *fp*, which are used as an input for the estimation model, consist of some imprecise attributes. In order to decrease any kind of vagueness related to the effort estimation, fuzzy logic can be inserted into the model and even though it does not guarantee to give the best result, acceptable results can be achieved.

This work in progress study, being an initial step in the development of a larger fuzzy effort estimation approach, aimed to seek an acceptable fuzzy estimation model based on fuzzy *fp*, by using fuzzy parameter estimates for clustered ISBSG data. Even though it is noticeable to conclude that the fuzzy effort estimation model can substitute the crisp estimation model, this undertaking was successful only with the clustered data in this study. Therefore, prior to estimations, one has to decide on the most appropriate cluster.

Based on these findings, the current research is on developing a fuzzy effort estimation model where the fuzzy effort function parameters and the fuzzy *fp* number are calculated based on the project attributes. Moreover, a more general fuzzy estimation model can be investigated for the whole data, independent of clusters. Additionally, the results of this study are efficient for crisp clustered data, whereas a project could be placed in a number of clusters at the same time. In such cases, fuzzy clustering methods can be used before estimating the project effort.

References

Jorgensen, M., Shepperd, M., “A systematic review of software development cost estimation studies,” IEEE Transactions on Software Engineering, vol. 33 (1), pp. 33-53, 2007.

Shepperd, M., Schofield, C., “Estimating software project effort using analogies,” IEEE Transactions on Software Engineering, vol. 23 (12), pp. 736-743, 1997.

Xu, Z., Khoshgoftaar, T.M., “Identification of fuzzy models of software cost estimation,” Fuzzy Sets and Systems, vol. 145 (1), pp. 141-163, 2004.

Musilek, P., Pedrycz, W., Succi, G., Reformat, M., “Software cost estimation with fuzzy models,” ACM SIGAPP Applied Computing Review, vol. 8 (2), pp. 24-29, 2000.

Idri, A., Alain, A., Khoshgoftaar, T.M., “Fuzzy analogy: a new approach for software

cost estimation," Proceedings of the International Workshop on Software Measurement, 2001.

Idri, A., Khoshgoftaar, T.M., Abran, A., "Investigating soft computing in case-based reasoning for software cost estimation," Engineering Intelligent Systems for Electrical Engineering and Communications, vol.10 (3), pp.147-158, 2002.

Azzeh, M., Neagu, D. Cowling, P.I., "Analogy-based software effort estimation using Fuzzy numbers," Journal of Systems and Software, vol. 84 (2), pp. 270-284, 2011.

Azzeh, M., Neagu, D., Cowling, P.I., "Fuzzy grey relational analysis for software effort estimation," Empirical Software Engineering, vol. 15 (1), pp. 60-90, 2010.

Lopez-Martin, C., Yanez-Marquez, C., Gutierrez-Tornes, A., "Predictive accuracy comarison of fuzzy models for software development effort of small programs," Journal of Systems and Software, vol. 81 (6), pp. 949-960, 2008.

Bansal, A., "Some Non Linear Arithmetic Operations on Triangular Fuzzy Numbers" *Advances in Fuzzy Mathematics, vol. 5* (2), pp. 147-156, 2010.

Aroba, J., Cuadrado-Gallego, J.J., Sicilia, M.A., Ramos, I., Garcia-Barriocanal, E., "Segmented software cost estimation models based on fuzzy clustering," Journal of Systems and Software, vol. 81, no. 11, pp. 1944-1950, 2008.

Kitchenham, B.A., Pickard, L.M., MacDonell, S.G., Shepperd, M.J., "What accuracy statistics really measure [software estimation]," Software, IEE Proceedings, vol. 148 (3), pp. 81-85, 2001