Interactive approach to release time problem of software under fuzzy environment

P. C. Jha\textsuperscript{1}, Indumati Singh\textsuperscript{1} and P. K. Kapur\textsuperscript{1}

\textsuperscript{1} Department of Operational Research, University of Delhi, Delhi-110007, India
E-mail: jhapc@yahoo.com
\textsuperscript{1} Department of Operational Research, University of Delhi, Delhi-110007, India
E-mail: indumati.singh@gmail.com
\textsuperscript{1} Department of Operational Research, University of Delhi, Delhi-110007, India
E-mail: pkkapur1@gmail.com

accepted July 17, 2010

Summary

Today’s era is an era of computers, software’s, technical up-gradations, and mechanization of processes by using different optimization techniques/strategies. Clients/Marketers in the software industries are now getting aware and are equally concerned about systematizing, strategizing, and optimizing their available resources. So one of their targets is now set to focus on finding out optimal time of launch of a software product in the market. Thus, to determine when to stop testing and release software in the market has become an important decision for the management. One of the techniques to quantify the release time is to formulate and solve a Software Release Time Optimization Problem. Due to several problems like system complexity, cutthroat competitions, varied market requirements and many other reasons, the software developers are able to provide ambiguous estimates on the available resources and their aspirations, which ultimately give rise to uncertainty (fuzziness) in the problem formulation. In such situations, crisp optimization technique may not serve the purpose to quantify the parameters. Defining the problem under fuzzy environment provides a better platform to quantify these uncertainties. In this paper, we have developed an interactive mathematical programming approach for a Bi-criterion Release Time Problem focusing on two conflicting objectives of Maximizing Reliability & Minimizing Cost simultaneously, on a flexible SRGM named Two Stage Yamada with Two types of Imperfect Debugging & Logistic Fault Removal Model. We discussed the problem with the help of numerical illustration.

Key words: Release time, Software Reliability Growth Model (SRGM), Software Release Time Decision (SRTD), fuzzy optimization, cost function, membership function.

1. INTRODUCTION

Today’s business world is client oriented, one who satisfy client requirements can sustain in the market effectively and profitably for long. In the field of computer systems Reliability and Cost are
two major concerns of client satisfaction. Both the constraints being measures of business require their quantification. Software reliability assessment techniques are of great importance to quantify the performance of a software system. The reliability of the software is quantified in terms of faults remaining in the software system. During the testing phase, the emphasis is on reducing the remaining fault content i.e., minimizing the failure intensity of faults in the software system, which consequently leads to increase in reliability of the software. The tools used to estimate the fault content, failure rate and fault removal rate per fault in software and to predict the reliability of the software at the time of its release are Software Reliability Growth Models (SRGM’s). Most of these SRGMs are NHPP (Non Homogeneous Poison Process) based and utilize historical failure data collected during the testing phase to evaluate the quality of the software. NHPP based SRGMs are generally classified into two groups. The first group contains models, which use the execution time (i.e., CPU time) or calendar time as a unit of fault detection period and are known as continuous time models [4, 12, 18] proposed under different set of assumptions. These models describe both exponential [4] and S-shaped Yamada [22] relationship between time/effort and the corresponding number of faults removed. The second group contains models, which use the test cases as a unit of fault detection/removal period. These models are called discrete time models, since the unit of software fault removal period is countable [5, 12, 15, 19]. A test case can be a single computer test run executed in an hour, days, weeks, or even months. Therefore, it includes the computer test run and length of time spent to visually inspect the software source code.

The utility of discrete SRGMs cannot be underestimated in spite of difficulties in terms of mathematical complexity involved. Discrete models are proposed regularly, as the software failure data sets are discrete, and these models many times provide better fit than their continuous time counter parts. Proposed exponential and modified exponential discrete SRGM in [22, 23] are assuming that the expected cumulative number of faults removed between the nth and the (n+1)th test cases is proportional to the number of faults remaining after the execution of the nth test run. They are also termed as homogeneous and non-homogeneous fault detection rate models. A Delayed S-shaped SRGM [7] is proposed in which the testing phase is assumed to have two different processes, namely, fault isolation and fault removal processes, the model is extended to the case when software contains several types of faults.

All the models proposed above have assumed the fault removal process (debugging) to be perfect i.e., every detected fault is removed with certainty. This assumption, however, seems to be a bit unrealistic. Due to the analytical nature of testing phase, manpower is mainly involved and hence there is a possibility that the testing team is not able to remove the fault perfectly on the detection of the failure and the original fault is remained or replaced by another fault. While the first phenomenon is known as imperfect debugging, the second is called fault generation. In recent years, several continuous SRGMs incorporating the concept of imperfect debugging and error generation have been proposed and studied [16, 6, 11, 25, 18, 19, 21]. Correspondingly discrete exponential and modified exponential SRGMs were proposed incorporating the concept of imperfect debugging [13]. In this paper we have used a discrete SRGM incorporating the effect of imperfect debugging and error generation with logistic fault removal [13].

However no software can be tested indefinitely in order to make it bug free owing to the prevailing paradox that software users requirement are conflicting with the developers. Software users want faster deliveries; cheaper software as well as quality product whereas software developers want to minimize their development cost, maximize the profit margins and meet the competitive requirements. Hence an imperative decision problem is to determine when to stop testing and release the software in the market known as ‘Software Release Time Decision Problem’ or SRTD problem. If the release of the software is unduly delayed, the manufacturer (software developer)
may suffer in terms of penalties and revenue loss, while a premature release may cost heavily in terms of fixes (removals) to be done after release, which consequently might harm manufacturer’s reputation. Release time problems have become a prime field of study for many eminent researchers [17, 24, 8, 9, 10, 14, 19].

The optimization problem of determining the optimal time of software release can be formulated based on goals set by the management in terms of cost, reliability and failure intensity etc. subject to the system constraints. An unconstrained release time optimization problem was first discussed with objectives of either cost minimization or reliability maximization [17]. Then a Release time problem was discussed with cost minimization objective under reliability aspiration constraint and reliability maximization objective under cost constraint [24]. The release time problems discussed above were formulated based on a single objective; however in the current scenario the developers have many conflicting objectives, therefore there is a need to incorporate the multiple criteria approach in the problem formulation and solution.

Bi-criterion release time optimization problem are the problems which deals with two simultaneous objectives. A problem of minimizing the total expected software cost and maximizing the reliability for exponential SRGM in continuous time is discussed in [9]. Such a release policy gives enough flexibility to the software developer to find out the optimal release time based on his priority with respect to reliability and cost components. If reliability is more important than higher weight may be applied to reliability objective as in case of safety critical projects. Similarly, for business application software packages, more weight may be applied to the cost objective. Bi-criterion release policy was determined for exponential and modified discrete SRGMs incorporating the concept of imperfect debugging [10]. Optimization techniques such as method of calculus, Mathematical Programming etc. are adopted to solve these problems.

In the existing research related to the software release time decision it is assumed that all the parameters of the problem are known precisely. Various objectives and restrictions are set by the management and cost coefficients involved in the cost function are determined based on past experience or from the previous available data base. This makes it difficult for the management to provide precise values of the various cost coefficients and objectives to be met by the release time. Moreover due to changing client specifications, lack of experience of testing team or novelty, changing testing environment due to innovations every now & then, complexity in the project involved, emerging factors unknowable at the start of the project adds imprecision and ambiguity to above mentioned definitions. It may also be possible that the management itself doesn’t set precise values in order to provide some tolerance on these parameters due to competitive conditions. All this leads to uncertainty (fuzziness) in the problem formulation. Crisp mathematical programming approaches provide no such mechanism to quantify these uncertainties. A fuzzy optimization approach is also introduced which is a flexible approach that permits a more adequate solution of such type of problems in the presence of vague information [2]. Now by usual application of fuzzy optimization approach we can provide client a single solution but by using interactive mathematical programming algorithm to solve the problem. By this approach we can provide client with multiple solutions. The main difference of this approach over the earlier release policies lies in degree of flexibility, in the sense that this approach provides the decision maker an opportunity to choose his/her most preferred solution amongst possible non-inferior solutions (Properly efficient solutions).

In this paper we have formulated a Bi-criterion release time optimization problem with Interactive approach to solve the problem of simultaneous dealing of two objectives of minimizing the expected software cost and maximizing the reliability subject under fuzzy environment (fuzzy objective, fuzzy inequalities in the constraints and problem parameters being fuzzy numbers) for
determining the optimal release time of the software and solved the problem by using fuzzy optimization technique. The release time problem we have formulated here is based on a continuous SRGM and the solution methodology is discussed with a numerical illustration. When a feasible solution of the problem exists we apply fuzzy optimization procedure to solve the problem, but in case we reach at the infeasibility, there we apply fuzzy goal programming optimization technique to provide a compromised solution of the same.

Rest of the paper is organized as follows: In section 2.1 we discussed the SRGM used to describe the functional relationship between failure phenomenon and time. In section 2.2 cost model used for the formulation is defined and then the problem is formulated in section 2.3. In section 3.1 we have discussed the basic concepts of fuzzy sets and presented an algorithm to solve the problem. Further in section 3.2 solution procedure is illustrated with a numerical example. Finally we have concluded the paper in section 4.

2. PROBLEM FORMULATION

2.1 Software Reliability Growth Model

In this paper we have considered the SRGM for error removal and failure phenomenon due to imperfect debugging and error generation model [13]. The SRGM and the optimization problem are based on the following notations.

<table>
<thead>
<tr>
<th>Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Initial number of faults in the software at the time when testing of software starts</td>
</tr>
<tr>
<td>a(t)</td>
<td>S-expected initial fault content at time t, a&gt;0</td>
</tr>
<tr>
<td>b(t)</td>
<td>Time dependent rate of fault removal per remaining faults</td>
</tr>
<tr>
<td>m(t)</td>
<td>Mean value function in the NHPP model, with m(0) = 0</td>
</tr>
<tr>
<td>m_r(t)</td>
<td>Expected number of faults removed by time t</td>
</tr>
<tr>
<td>m_d(t)</td>
<td>Expected number of faults detected by time t</td>
</tr>
<tr>
<td>P</td>
<td>The probability of perfect debugging of fault removal on a failure</td>
</tr>
<tr>
<td>α</td>
<td>Constant rate of error generation</td>
</tr>
<tr>
<td>β</td>
<td>Constant parameter in the logistic function</td>
</tr>
<tr>
<td>R(x</td>
<td>T)</td>
</tr>
<tr>
<td>C_1(C_2)</td>
<td>Cost incurred on a perfect (imperfect) debugging effort before release of the software system</td>
</tr>
<tr>
<td>C_3(C_4)</td>
<td>Cost incurred on a perfect (imperfect) debugging effort after release of the software system. (C_3 &gt; C_1, C_4 &gt; C_2)</td>
</tr>
<tr>
<td>C</td>
<td>Testing cost per unit time</td>
</tr>
<tr>
<td>C_0</td>
<td>Total available resources</td>
</tr>
<tr>
<td>T_w</td>
<td>Warranty period</td>
</tr>
<tr>
<td>T</td>
<td>Release time of the software</td>
</tr>
<tr>
<td>T*</td>
<td>Optimal release time</td>
</tr>
<tr>
<td>R_0</td>
<td>Desired level of software reliability at the release time (0 &lt; R_0 &lt; 1)</td>
</tr>
</tbody>
</table>
given as
\[ \frac{dm_r(t)}{dt} = pb(t)[a(t) - m_r(t)] \quad \& \quad \frac{dm_f(t)}{dt} = b(t)[a(t) - m_f(t)]. \quad (1) \]

where, \( b(t) = b \left[ \frac{1}{1 + \beta e^{-bt}} - \frac{1}{1 + \beta + bt} \right] \), and \( a(t) = a + \alpha m_r(t). \quad (2) \)

Solving the above differential equation (2) under initial condition \( m_r(0) = 0 \), we get mean value

function as \( m_r(t) = \frac{a}{1-\alpha} \left[ 1 - \left( \frac{(1 + \beta + bt) e^{-bt}}{1 + \beta e^{-bt}} \right)^{p(1-\alpha)} \right]. \quad (3) \)

Here, \( m_r(t) = \frac{a}{(1-\alpha)} \). When \( t \) tends to \( \infty \), which implies that if testing is carried out for an infinite time more faults are removed as compared to the initial fault content because there are some errors added to the software due to error generation.

Relationship between \( m_r(t) \) and \( m_f(t) \) is given by \( m_r(t) = pm_f(t) \quad (4) \)

Therefore the failure phenomenon \( m_f(t) \) is given as

\[ m_f(t) = \frac{a}{p(1-\alpha)} \left[ 1 - \left( \frac{(1 + \beta + bt) e^{-bt}}{1 + \beta e^{-bt}} \right)^{p(1-\alpha)} \right]. \quad (5) \]

The reliability of the software is given as

\[ R(T|T_W) = e^{-\left( m_f(T+T_W) - m_f(T) \right)}. \quad (6) \]

2.2 The Cost Model

Release time optimization problem requires the definition of cost function. The cost function earlier proposed includes the cost of testing per unit time, cost of removing a fault during testing and operation phase [17]. The effect of imperfect debugging and fault generation on the cost function was discussed in [14]. They discussed that the parameters \( p \) and \( \alpha \) are influenced by a number of factors, such as the experience of the testing personnel, the testing strategy adopted etc. Employing skilled professionals and using well established testing methodologies can improve the testing efficiency, but this has to be achieved at a higher testing cost. Therefore, cost per unit testing time varies with the testing efficiency and is a monotonically increasing function of \( p \) and \( (1- \alpha) \). Moreover the cost of debugging a fault is different for both perfect debugging and imperfect debugging. A reasonably realistic approach of warranty cost is considered. The cost of failure and removal of a fault during a fixed warranty period after the release of the software is included. The software cost model during the software life cycle is considered as

\[ C(T) = (C_1 p + C_2 (1-p)) m_f(t) + (C_3 p + C_4 (1-p)) \left( m_f(\infty) - m_f(t) \right) + \frac{CT}{(1-p(1-\alpha))}. \quad (7) \]
We have considered all these parameters to be fuzzy numbers. The fuzzy cost function can be given as
\[
\tilde{C}(T) = \left(\tilde{C}_1 p + \tilde{C}_2 (1 - p)\right)m_f(T) + \left(\tilde{C}_3 p + \tilde{C}_4 (1 - p)\right)\left(m_f(T + T_w) - m_f(T)\right) + \frac{\tilde{C}_T}{(1 - p(1 - \alpha))}.
\]
\(~\) on the parameters represents that they are fuzzy numbers. These parameters are assumed to be Triangular Fuzzy Numbers (TFN).

### 2.3 Problem formulation

Using the fuzzy cost function given by equation (8) the release time problem of simultaneously dealing of two objectives, minimizing the fuzzifier cost function, maximizing the fuzzifier reliability is formulated as follows:-

Minimize \( \tilde{C}(T) \)

Maximize \( \tilde{R}(x/T) \)
\( T \geq 0 \)

\( (P1) \)

### 3. PROBLEM SOLUTION

#### 3.1 Preliminaries and Solution Methodology

This section discusses some of the basic concepts of the fuzzy set theory (FST) and fuzzy Mathematical Programming [1].

**Definition 1. Fuzzy set:** Let \( X \) be the universe whose generic element is denoted by \( x \). A fuzzy set \( A \) in \( X \) is a function \( A : X \rightarrow [0, 1] \).

**Definition 2. \( \alpha \)-cut:** The \( \alpha \)-cut of the fuzzy set \( A \) in \( X \) is the crisp set \( A_\alpha \) given by \( A_\alpha = \{ x \in X : \mu_A(x) > \alpha \} \) where \( \alpha \in (0, 1) \).

**Definition 3. Fuzzy number:** A fuzzy set \( A \) in \( \mathbb{R} \) is called a fuzzy number if it satisfies the following conditions:-

(i) \( A \) is normal.
(ii) \( A \) is convex.
(iii) \( \mu_A \) is upper semi continuous.
(iv) Support of \( A \) is bounded.

**Definition 4. Triangular fuzzy number (TFN):** A fuzzy number \( A \) denoted by the triplet \( A = (a_1, a, a_u) \) having the shape of a triangle is called a TFN. The \( \alpha \)-cut of a TFN is the closed interval \( A_\alpha = [a_{1\alpha}, a_{u\alpha}] = [(a-a_1)\alpha + a_1, (a-a_u)\alpha + a_u] \), \( \alpha \in (0, 1) \).

**Theorem 1.** Let \( A \) be a fuzzy set in \( \mathbb{R} \). Then \( A \) is a fuzzy number if and only if there exists a closed interval (which may be singleton) \( [a, b] \neq \emptyset \) such that \( \mu_A(x) = \begin{cases} \{1, x \in [a,b] \} & \text{where} \\
\{l(x), x \in (-\infty, a] \} & l(x) = 0 \text{ for } x \in (-\infty, w_1), w_1 < a \end{cases} \)

(i) \( l : (-\infty, a) \rightarrow [0, 1] \) is non-decreasing, continuous from the right and
(ii) \( r : (b, \infty) \to [0, 1] \) is non increasing, continuous from the left and \( r(x) = 0 \) for \( x \in (w_2, \infty), w_2 > b \) and \( \mu_A(x) \) is called ‘Membership Function’ of fuzzy set A on \( \mathbb{R} \).

An element mapping to the value 0 means that the member is not included in the given set, 1 describes a fully included member. Values strictly between 0 and 1 characterize the fuzzy members. Figure 1 illustrates a fuzzy set graphically.

Next we define the ranking of fuzzy numbers. Ranking of fuzzy number is an important issue in the study of fuzzy set theory and is useful in various applications. Fuzzy mathematical programming is one of the applications. We use the Ranking function (index) approach for ranking the fuzzy numbers for our problem and the same is illustrated below.

Ranking function (index) approach [27]

Let \( N(\mathbb{R}) \) be the set of all fuzzy numbers in \( \mathbb{R} \) and \( A, B \in N(\mathbb{R}) \). Define a function \( F : N(\mathbb{R}) \to \mathbb{R} \), called a ranking function or ranking index, where \( F(A) \leq F(B) \) is equivalent to \( A \leq B \). Following indices are proposed by [27].

(i) \( F_1(A) = \left( \int_{a_1}^{a_u} x \mu_A(x) dx \right) / \left( \int_{a_1}^{a_u} \mu_A(x) dx \right) \), where \( a_1 \) and \( a_u \) are the lower and upper limits of the support of A. The value \( F_1(A) \) is the centroid of the fuzzy number \( A \in N(\mathbb{R}) \). For example, If \( A = (a_1, a, a_u) \) is a triangular fuzzy number (TFN) where \( a_1 \) and \( a_u \) are the lower and upper limits of the support of A and \( a \) is the model value then \( F_1(A) = (a_1 + a + a_u) / 3 \).

(ii) \( F_2(A) = \left( \int_0^{a_{\max}} m \left[ a_{L\alpha}, a_{R\alpha} \right] d\alpha \right) \), Where \( a_{\max} \) is the height of A, \( A_{\alpha} = \left[ a_{L\alpha}, a_{R\alpha} \right] \) is a \( \alpha \)-cut, \( \alpha \in (0, 1] \), and \( m \left[ a_{L\alpha}, a_{R\alpha} \right] \) is the mean value of elements of the \( \alpha \)-cut. For example, If \( A = (a_1, a, a_u) \) is a TFN, \( a_{\max} = 1 \) and \( A_{\alpha} = \left[ a_{L\alpha}, a_{R\alpha} \right] = [(a-a_1)\alpha + a_1, (a-a_u)\alpha + a_u] \) then \( m \left[ a_{L\alpha}, a_{R\alpha} \right] = ((2a-a_1-a_u)\alpha + (a_1+a_u))/2 \) and \( F_2(A) = (a_1 + 2a + a_u) / 4 \).

The following algorithm specifies the sequential steps to solve the fuzzy mathematical problems formulated in section 3.2.
Algorithm [26]

1. Compute the crisp equivalent of the fuzzy parameters using a defuzzification function (ranking of fuzzy numbers). Same defuzzification function is to be used for each of the parameters. Here we use the defuzzification function of type $F_2(A) = \left( a_1 + 2a + a_0 \right)/4$.

2. Incorporate the objective function of the fuzzifier min (max) as a fuzzy constraint with a restriction (aspiration) level.

3. Define appropriate membership functions for fuzzy inequalities. The membership function for the fuzzy less than or equal to and greater than or equal to type are given as

$$
\mu_i(T) = \begin{cases} 
\frac{1}{G^* - G_i(T)} ; & G_i(T) \leq G_0 \\
\frac{G_i(T) - G_0}{G^* - G_0} ; & G_0 < G_i(T) \leq G^*
\end{cases}
$$

respectively.

Where $G_0$ and $Q_0$ are the restriction and aspiration levels respectively and $G^*$ and $Q^*$ are the corresponding the tolerance levels. The membership functions can be a linear or piecewise linear function that is concave or quasiconcave.

4. Employ the Extension principle to identify the fuzzy decision [Bector & Chandra, 2005], which results in a crisp mathematical programming problem given by:

Maximize $\alpha$

Subject to $\mu_i(T) \geq \alpha_i$, $i = 1, 2, \ldots, n$;

$\alpha \geq 0$, $\alpha \leq 1$, $T \geq 0$

and can be solved by the standard crisp mathematical programming algorithms.

5. Solving the problem following steps 1-4, objective of the problem is also treated as a constraint. In the release time decision problem under consideration each constraint actually corresponds to an optimization objective. Hence we can consider each constraint to be an objective for the decision maker for which imprecise restrictions (aspirations) are specified by the client/developer. The problem can be looked as a fuzzy multiple objective mathematical programming problem. Further each objective can have different level of importance and can be assigned weight to measure the relative importance. The resulting problem can be solved by the weighted min max approach. The crisp formulation of the weighted problem is given as

Maximize $\alpha$

Subject to $\mu_i(T) = w_i \alpha_i$, $i = 1, 2, \ldots, n$

$\alpha \geq 0$, $T \geq 0$, $\sum_{i=1}^{n} w_i = 1$

where $n$ is the number of constraints in $P^{**}$ and $\alpha$ represents the degree upto, which the aspiration of the decision maker is met. The problem $P^{**}$ can also be solved using standard mathematical programming approach.

Interactive Algorithm [28]

The following algorithm summarizes the Interactive weighted sum approach termed as interactive mathematical programming approach to solve the above problem.
Notations

l : number of properly efficient solutions and their respective criterion vectors to be present to
decision maker at each iteration.
w : final iteration $[l_k, \mu_k]$ interval width.
t : the number of iterations to be performed.
\( r \) : a reduction factor.

**Step 1.** Specify parameters \( r \) and \( t \). Let the iteration counter \( h = 1 \).

**Step 2.** Generate \( 10 \times K \) weighing vectors randomly from

\[
\Lambda^h = \left\{ \lambda \in R^K \mid \lambda_k \in \left[ t_k^h, \mu_k^h \right], \sum_{k=0}^K \lambda_k = 1 \right\}.
\]

Filter \( 10 \times K \) weighting vectors to get \( 2 \times l \) most different weighting vectors.

**Step 3.** Solve \( 2 \times l \) release problems using \( 2 \times l \) weighting vectors. For each problem we get a properly efficient solution based on Bi-criterion Release policies.

**Step 4.** Obtain \( 2 \times l \) criterion vectors for each of \( 2 \times l \) properly efficient solution. Filter \( 2 \times l \) criterion vectors to get \( l \) most different criterion vectors and also list corresponding properly efficient solutions.

**Step 5.** Present \( l \) most different criterion vectors and their respective properly efficient solutions and obtain the decision maker’s most preferred criterion vector and its corresponding properly efficient solutions.

**Step 6.** Perform reverse filtering using the decision maker’s most preferred criterion vector as seed point and obtain the closet criterion vectors and list corresponding properly efficient solution.

**Step 7.** Present closet criterion vectors and their respective properly efficient solutions to the decision maker and obtain the decision maker’s most preferred criterion vector and its corresponding properly efficient solution. Designate it as \((Z^h, X^h)\).

**Step 8.** If iteration number is less than the prior fixed number of iterations, (i.e. \( h < t \)), set \( h = h+1 \) and form.

\[
\Lambda^{h+1} = \left\{ \lambda \in R_K \mid \lambda_k \in \left[ t_k^{h+1}, \mu_k^{h+1} \right], \sum_{k=0}^K \lambda_k = 1 \right\}.
\]

a more concentrated weighting vector space.

Where

\[
\begin{bmatrix} t_k^{h+1}, \mu_k^{h+1} \end{bmatrix} = \begin{cases} \left[ 0, r^h \right], & \text{if } \lambda_k^h - \frac{r^h}{2} \leq 0 \\ \left[ 1-r^h, 1 \right], & \text{if } \lambda_k^h + \frac{r^h}{2} \geq 1 \\ \left[ \lambda_k^h - \frac{r^h}{2}, \lambda_k^h + \frac{r^h}{2} \right], & \text{otherwise} \end{cases}
\]

And go to step 1.
Otherwise stop with \((Z^h, X^h)\) as a final solution.

In the next section we are using both the algorithms to find optimal release time of the software corresponding to the above formulated problem with a numerical example.

### 3.2 Numerical Example

In this section, first we explain the fuzzy mathematical programming approach using the algorithm discussed above to find the release time of the software. First we estimate the unknown parameters of the mean value function of the failure phenomenon of the imperfect debugging and error generation model in equation (3). For this we use the software failure data set from a real time command and control system data reported by Brooks and Motley to estimate model parameters. The data set we are using here is of software which is tested for 35 months of CPU time during which 1301 failures were reported. Parameters are estimated using the non-linear regression function of SPSS software. Estimated values of parameters and residual sum of square \((R^2)\) are given in Table 1.

#### Table 1. Estimated values of parameters and residual sum of square \((R^2)\)

<table>
<thead>
<tr>
<th>DS-I</th>
<th>Estimates of Parameters</th>
<th>Comparison Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>1350.567358</td>
<td>0.175327361</td>
<td>0.009</td>
</tr>
</tbody>
</table>

The various cost parameters \(C_1, C_2, C_3, C_4\) and \(C\) total available resources \(C_0\) and expected reliability \(R_0\) are the triangular fuzzy numbers represented as \(A=(a_1,a,a_u)\) in Table 2. The value of these fuzzy numbers are specified by the management based on the past experience and/or expert opinion. We assume all these fuzzy parameters values are known and are given in Table 2. Using the defuzzification function \(F_2(A) = (a_1 + 2a + a_u)/4\) we defuzzify these fuzzy numbers. Defuzzified values of these parameters are also given in Table 2.

#### Table 2. Defuzzified values of these parameters

<table>
<thead>
<tr>
<th>Fuzzy Parameter (A)</th>
<th>a₁</th>
<th>a</th>
<th>a_u</th>
<th>Defuzzified value ((F_2(A)))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_1)</td>
<td>6.4</td>
<td>6.8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>(C_2)</td>
<td>5.4</td>
<td>5.8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>(C_3)</td>
<td>27</td>
<td>27.5</td>
<td>30</td>
<td>28</td>
</tr>
<tr>
<td>(C_4)</td>
<td>21</td>
<td>24.6</td>
<td>25.8</td>
<td>24</td>
</tr>
<tr>
<td>(C)</td>
<td>10</td>
<td>11.2</td>
<td>11.6</td>
<td>11</td>
</tr>
<tr>
<td>(C_0)</td>
<td>38000</td>
<td>39000</td>
<td>44000</td>
<td>40000</td>
</tr>
<tr>
<td>(R_0)</td>
<td>0.9</td>
<td>0.91</td>
<td>0.96</td>
<td>0.92</td>
</tr>
</tbody>
</table>

The tolerance level of cost is \(C^* = 55000\), \(R^* = 0.86\) and the warranty period \(T_w = 18\) months. Now we restate the problem \(P1\) using the defuzzification function \(F_2(A)\).

\[
\begin{align*}
\text{Minimize} & \quad F_2(\tilde{C}(T)) \\
\text{Maximize} & \quad F_2(\tilde{R}(x/T))
\end{align*}
\]

\(T \geq 0\) \quad (P2)
Using the values of the fuzzy parameters as given in Table 2 and substituting in the defuzzification function $F_2(A)$ to obtain the defuzzified values of these parameters. Hence the problem $P_2$ is rewritten as:

\[
\text{Minimize } C(T) = (7^*0.99 + 6^*[1-0.99])m_r(T) + (28^*0.99 + 24^*[1-0.99])(m_r(T+18) - m_r(T)) + \frac{11^*T}{(1-0.99(1-0.009))}
\]

\[
\text{Maximize } R(T_W | T) = e^{-(m_r(T+18) - m_r(18))}
\]

\[T \geq 0\]  

(P3)

Where

\[
m_r(T) = \frac{1350.567358}{1-0.099} \left[ 1 - \left( \frac{1 + 7.397348702 + 0.175327361^*T} {1 + 7.397348702e^{-0.175327361^*T}} \right)^{0.99(1-0.009)} \right]
\]

\[
\mu_i(T) = \begin{cases} 
1 & : C(T) \leq 40000 \\
\frac{55000 - (7^*0.99 + 6^*[1-0.99])m_r(T) + (28^*0.99 + 24^*[1-0.99])(m_r(T+18) - m_r(T)) + 11^*T}{55000 - 40000} & : 40000 < C(T) \leq 55000 \\
\frac{55000}{40000} & : C(T) > 55000
\end{cases}
\]

\[
m_r(T+18) = \frac{1350.567358}{1-0.099} \left[ 1 - \left( \frac{1 + 7.397348702 + 0.175327361^*(T+18)} {1 + 7.397348702e^{-0.175327361^*(T+18)}} \right)^{0.99(1-0.009)} \right]
\]

Now we incorporate the fuzzifier $\min \text{ objective as a restriction level constraint with minimum cost function specified as a TFN } \bar{C}_0 = (38000, 39000, 44000) , \bar{R}_0 = (0.9, 0.91, 0.96) , \text{ and upper tolerance level } C^* = 55000 , R^* = 0.86 \text{. Using the defuzzification function } F_2(A) \text{ we obtain } F_2(\bar{C}_0) = C_0 = 40000 , F_2(\bar{R}_0) = R_0 = 0.92 \text{. The problem } P_3 \text{ can now be restated for finding } T \text{ as follows:}

\text{Find } T \\
\text{Subject to } \\
C(T) \leq 40000, \quad R(T | T) \geq 0.92 \\
T \geq 0 \]

(P4)

We define the following membership functions $\mu_i(T) ; i = 1, 2 \text{ for each of the fuzzy inequalities in } P_4.$

\[
\mu_1(T) = \begin{cases} 
1 & : C(T) \leq 40000 \\
\frac{0.92 - e^{-m_r(T+18) - m_r(T)}}{0.92 - 0.86} & : 0.86 < R(T_W | T) \leq 0.92 \\
0 & : R(T_W | T) < 0.86
\end{cases}
\]

\[
\mu_2(T) = \begin{cases} 
1 & : R(T_W | T) \geq 0.92 \\
0 & : R(T_W | T) < 0.86
\end{cases}
\]

Now we formulate the crisp optimization problem $P_5 \text{ using the above membership functions of equation (10), to identify the fuzzy decision and solve the fuzzy system of inequalities corresponding to the problem } P_4.$
Maximize $\alpha$
Subject to

$$\mu_1(T) = \left(7\times 0.99 + 6\times (1-0.99) \right)m_1(T) + \left(28\times 0.99 + 24\times (1-0.99) \right)(m_1(T)+18) - m_1(T) + \frac{11^T}{1-0.99(1-0.99)} \geq \alpha$$

$$\mu_2(T) = \frac{0.92 - e^{-\left(m_r(t+18) - m_r(t)\right)}}{0.92 - 0.86} \geq \alpha$$

$\alpha \geq 0$, $\alpha \leq 1$, $T \geq 0$  \hspace{1cm} (P5)

The constraints corresponding to cost & reliability in the problem P4 and P5 are two important objectives in the SRTD problem which may be assigned different weights as per their relative importance. Now the problem gets converted to problem P6.

Maximize $\alpha$
Subject to

$\mu_1(T) \geq w_1 \alpha$ \hspace{1cm} i=1, 2;

$\mu_2(T) \geq w_2 \alpha$

$w_1 + w_2 = 1$  \hspace{1cm} (P6)

$\alpha \geq 0$, $\alpha \leq 1$, $T \geq 0$

Problem P6 can be solved by using standard mathematical programming approach in LINGO [13] software. Here we are attaching different weights $w = \{w_1, w_2\}$ to both constraints of P6 see Table 3.

<table>
<thead>
<tr>
<th>Weights (w1,w2)</th>
<th>t(weeks)</th>
<th>Cost ($)</th>
<th>R(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.1,0.9)</td>
<td>64.98</td>
<td>47328.28</td>
<td>0.863468232</td>
</tr>
<tr>
<td>(0.2,0.8)</td>
<td>65.2</td>
<td>47456.14</td>
<td>0.868187824</td>
</tr>
<tr>
<td>(0.3,0.7)</td>
<td>65.42</td>
<td>47456.14</td>
<td>0.868187824</td>
</tr>
<tr>
<td>(0.4,0.6)</td>
<td>65.74</td>
<td>47770</td>
<td>0.8797140779</td>
</tr>
<tr>
<td>(0.5,0.5)</td>
<td>65.749</td>
<td>47775.23</td>
<td>0.879315959</td>
</tr>
<tr>
<td>(0.6,0.4)</td>
<td>66.87</td>
<td>48426.86</td>
<td>0.899373041</td>
</tr>
</tbody>
</table>

Suppose the decision maker choice is the solution corresponding to (w1=0.2, w2=0.8). Now using the reduction method we find the new weighting vector space and new vector space is formed using $0.23 \leq w_1 \leq 0.25$ and $0.77 \leq w_2 \leq 0.75$. The previous procedure is repeated again. The six most distinct solutions are calculated using the new vector spaces. The three most different solutions presented to the decision makers after filtering are corresponding to the weighting vectors (w1,w2)={(0.235,0.765), (0.24,0.76), (0.25,0.75)} in Table 4.

<table>
<thead>
<tr>
<th>Weights (w1,w2)</th>
<th>t(weeks)</th>
<th>Cost ($)</th>
<th>R(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.235,0.765)</td>
<td>65.23</td>
<td>47473.58</td>
<td>0.868819628</td>
</tr>
<tr>
<td>(0.24,0.76)</td>
<td>65.27</td>
<td>47496.83</td>
<td>0.869657687</td>
</tr>
<tr>
<td>(0.25,0.75)</td>
<td>65.29</td>
<td>47508.45</td>
<td>0.87007486</td>
</tr>
</tbody>
</table>

Again using the reduction method we find the new weighting vector space and new vector space is formed using $0.15 \leq w_1 \leq 0.19$ and $0.85 \leq w_2 \leq 0.81$. The previous procedure is repeated again. The
six most distinct solutions are calculated using the new vector spaces. The three most different solutions presented to the decision makers after filtering are corresponding to the weighting vectors \((w_1,w_2)\)={(0.15,0.85), (0.16,0.84), (0.18,0.82)} in Table 5.

**Table 5. The three most different solutions**

<table>
<thead>
<tr>
<th>Weights (w1,w2)</th>
<th>t(weeks)</th>
<th>Cost ($)</th>
<th>R(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.15,0.85)</td>
<td>65.07</td>
<td>47380.59</td>
<td>0.865417462</td>
</tr>
<tr>
<td>(0.16,0.84)</td>
<td>65.09</td>
<td>47392.21</td>
<td>0.865847133</td>
</tr>
<tr>
<td>(0.18,0.82)</td>
<td>65.13</td>
<td>47415.46</td>
<td>0.866702688</td>
</tr>
</tbody>
</table>

Suppose the decision maker choice is the solution corresponding to \((w_1=0.4, w_2=0.6)\). Now using the reduction method we find the new weighting vector space and new vector space is formed using \(0.45 \leq w_1 \leq 0.5\) and \(0.55 \leq w_2 \leq 0.5\). The previous procedure is repeated again. The six most distinct solutions are calculated using the new vector spaces. The three most different solutions presented to the decision makers after filtering are corresponding to the weighting vectors \((w_1,w_2)\)={(0.44,0.56), (0.45,0.55), (0.46,0.54)} in Table 6.

**Table 6. The three most different solutions**

<table>
<thead>
<tr>
<th>Weights (w1,w2)</th>
<th>t(weeks)</th>
<th>Cost ($)</th>
<th>R(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.45,0.55)</td>
<td>65.95</td>
<td>47892.066</td>
<td>0.883167273</td>
</tr>
<tr>
<td>(0.46,0.54)</td>
<td>66</td>
<td>47921.13</td>
<td>0.884107361</td>
</tr>
<tr>
<td>(0.5,0.5)</td>
<td>65.749</td>
<td>47775.23</td>
<td>0.879315959</td>
</tr>
</tbody>
</table>

Again using the reduction method we find the new weighting vector space and new vector space is formed using \(0.47 \leq w_1 \leq 0.52\) and \(0.53 \leq w_2 \leq 0.48\). The previous procedure is repeated again. The six most distinct solutions are calculated using the new vector spaces. The three most different solutions presented to the decision makers after filtering are corresponding to the weighting vectors \((w_1,w_2)\)={(0.47,0.53), (0.48,0.52), (0.49,0.51)} in Table 7.

**Table 7. The three most different solutions**

<table>
<thead>
<tr>
<th>Weights (w1,w2)</th>
<th>t(weeks)</th>
<th>Cost ($)</th>
<th>R(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.47,0.53)</td>
<td>66.05</td>
<td>47950.19</td>
<td>0.885040387</td>
</tr>
<tr>
<td>(0.48,0.52)</td>
<td>66.1</td>
<td>47979.26</td>
<td>0.885966396</td>
</tr>
<tr>
<td>(0.49,0.51)</td>
<td>66.12</td>
<td>47990.88</td>
<td>0.886334845</td>
</tr>
</tbody>
</table>

Here it can be noted that the fuzzy optimization method provides sub-optimal solutions due its subjective nature. Table 6 and Table 7 are showing different set of solutions.

4. CONCLUSIONS

In this paper we have formulated a bi-criterion fuzzy software release time decision problem of simultaneously minimizing the cost function & maximizing the reliability and discussed the fuzzy mathematical programming procedure along with an interactive mathematical programming.
approach for providing higher flexibility and multiple choices of output to the client. An optimization problem is illustrated with the help of a numerical example for number of feasible solutions with respect to different weights attached to the objectives as per client’s interest. So after presenting these solutions to client, client would be able to choose best feasible and optimal solution for the problem as per his requirement. However since the method provides huge amount of flexibility to the management in decision making of inputs under fuzzy environment, and provide more than single output domain to the client, so has wider application despite a lot of subjectivity in its nature. This is an interesting topic of further studies in fuzzy optimization.

REFERENCES


