Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Software effort estimation based on the optimal Bayesian belief network

Fatemeh Zare*, Hasan Khademi Zare, Mohammad Saber Fallahnezhad

Department of Industrial Engineering, Yazd University, Yazd, Iran

ARTICLE INFO

Article history:
Received 6 December 2015
Received in revised form 29 May 2016
Accepted 2 August 2016
Available online xxx

Keywords:
Software effort estimation
Bayesian belief network
Optimal control
Software quality

ABSTRACT

In this paper, we present a model for software effort (person-month) estimation based on three levels Bayesian network and 15 components of COCOMO and software size. The Bayesian network works with discrete intervals for nodes. However, we consider the intervals of all nodes of network as fuzzy numbers. Also, we obtain the optimal updating coefficient of effort estimation based on the concept of optimal control using Genetic algorithm and Particle swarm optimization for the COCOMO NASA database. In the other words, estimated value of effort is modified by determining the optimal coefficient. Also, we estimate the software effort with considering software quality in terms of the number of defects which is detected and removed in three steps of requirements specification, design and coding. If the number of defects is more than the specified threshold then the model is returned to the current step and an additional effort is added to the estimated effort. The results of model indicate that optimal updating coefficient obtained by genetic algorithm increases the accuracy of estimation significantly. Also, results of comparing the proposed model with the other ones indicate that the accuracy of the model is more than the other models.

1. Introduction

Software cost estimation is the process of estimating the needed effort for development of a software system. The important issue in the software cost estimation is the accuracy of estimation values. However, many models are suggested in this context, but the accuracy of estimation is still a challenging issue [1]. There are many uncertain variables in the software effort estimation. For example, several factors such as development method, organization and programing language make the software size to be a fuzzy number [2]. Also, many important variables in estimating the needed effort are determined exactly at the end of the project. Thus, they should be estimated before implementing the project [3]. The accurate cost estimation can help managers categorize and prioritize software development projects with respect to the general plan of the organization [1] and also, assign resources properly [4]. Also, the accurate software cost and time estimation is important in the negotiation of contracts, tenders, scheduling, control and monitoring of the projects. Underestimation can lead to lack of enough budget and not completing the software project on time and overestimation can lead to not winning the tender and missing the project [1].

There are different methods for the effort estimation of software development. The main important methods can be categorized as model-based and expert-based methods. Model-based methods are categorized as statistic-based methods like regression and intelligent-based ones like machine learning methods [5]. Todays, the attention to the machine learning methods is increasing because they are capable of modelling the relation between effort needed for software development and its components. These methods can be categorized as Neutral networks, Bayesian networks, and Genetic algorithms [6,7].

One of the machine learning methods is the Bayesian belief network. A Bayesian network has a graph and probability tables which the nodes indicate the probabilistic variables and the arcs indicate the cause and effect relations among these variables. Theses probabilities for without- parent nodes show marginal probabilities and for nodes with parent show conditional probabilities. Almost, all of the Bayesian networks are built from gathered data and expert opinions. Using Bayes theory, the prior probabilities are converted to posterior ones which this conversion is known as learning process. Posterior probability distributions are determined based on the prior distribution and the conditional probabilities. Usually, prior distributions are determined by experts and conditional probabilities are obtained by datasets [8,9]. Cause and effect relations of

* Corresponding author.
E-mail addresses: fatemezarebaghi@yahoo.com (F. Zare), Hkhademiz@yazd.ac.ir (H. Khademi Zare), fallahnezhad@yazd.ac.ir (M.S. Fallahnezhad).

http://dx.doi.org/10.1016/j.asoc.2016.08.004
1568-4946/© 2016 Elsevier B.V. All rights reserved.

Bayesian network allow experts to complete the data structure and keep them available. Also, they provide more flexibility for decision support system based on available data [8]. The effort needed for the software development is a continuous variable and Bayesian network needs discrete variables. Thus, effort estimation must be approximated by some independent intervals [10]. The selection of discretizing method and the number of discretized intervals is effective on the estimation accuracy [11]. Bayesian Belief Network is appropriate for small and incomplete datasets. Also, it allows structural learning and combines different knowledge resources. However, Bayesian network is an acyclic graph and doesn't cover feedback loops that are sometimes useful in modelling [12]. Stamelos et al. [13] presented a three level Bayesian network to estimate the software productivity (DSI/MM[1]) based on COCOMO components in which the first level is deterministic and second and third level are probabilistic but they didn't apply optimal control approach. Parag et al. [10] suggested a bi-level Bayesian network to estimate the software effort based on three components including the software development method, tools and tool experience. Also, the estimation is updated by considering new information such as expert opinions in the model. Bibi et al. [14] applied the Bayesian network to estimate the effort for software development based on the phases of the software development process. The prominent advantage of Bayesian network in their model was the ability of simplifying the processes such as planning, control and operational management. Akhtar [1] suggested two steps to estimate the software effort. First, he specified the effective components on the software cost. Then, each component is decomposed into some variables to quantify these components so that the experts can determine the values of them by questionnaire. Finally, Ant colony algorithm is used to obtain the best probability result for each component. In the second step, a Bayesian cause and effect relation is built among these components through combining the probabilities obtained in the previous step and obtaining the overhead cost. This overhead cost can be used to obtain the productivity and the effort estimation. Fuentetaja et al. [3] suggested a multi-stage process to build a semi-automated Bayesian network for estimating the effort for software development using data mining tool such as WEKA. In this research, first an automated model was built based on six phases of software development including proposal, requirements specification, design, coding and construction, installation and end of project. Second, this model was adjusted by the experts. This process was designed to estimate the effort, quality and risk.

Khatibi et al. [15] used Analogy-based method to estimate the effort needed for the software development. They also applied Particle swarm optimization (PSO) to control this model optimally and obtained the optimal weights of features related to the similarity function. The objective function of PSO was to minimize the defined error measurement. COCOMO method is an estimation method that is based on components rather than development process. COCOMOII is a developed version of COCOMO which uses five factors as exponential components and seventeen factors as effort coefficients [16]. Soleimani et al. used PSO algorithm to optimize error of dataset projects estimations and they estimated the optimal values of a and b that are parameters of COCOMO ($E = a * size^b$) to obtain the best accuracy. They used basic COCOMO and didn't consider the effort components of COCOMO or COCOMOII. In the other study [4], Genetic algorithm was applied to tune the parameters of COCOMOII and the optimal value of the effort needed for the software development was obtained. Also, COCOMOII method was modified by the introduction of some other parameters for increasing the accuracy of estimation.

In this research, we present a model to estimate the software effort (man-month) based on three level Bayesian network built from fifteen COCOMO components and the software size. Also, the optimal updating coefficient of the effort estimation is obtained by Genetic and PSO algorithm using COCOMO NASA dataset. The obtained effort is modified by the optimal coefficient. Also, the quality of developed software is considered in terms of the number of defects detected and removed in three steps of the requirements specification, design and coding. If the number of defects exceeds than the defined threshold for the defects then the estimation process will be returned to the current step and an additional effort is added to the estimated effort. The proposed method has following advantages:

- We use Bayesian network which is based on the cause and effect relations and also, it's not necessary to have complete information and missing data can be estimated [12].
- As shown in Table 1, the application of Bayesian belief network was not investigated in recent years.
- The Bayesian belief network works with discrete intervals. However, we suggest that the intervals of nodes in three levels of Bayesian network except effort node is as fuzzy numbers instead of crisp discrete ones. Thus, the intervals of nodes have common terms. This issue has not been addressed before and provides more flexibility. Also, this approach increases accuracy of estimation because of decreasing the discretization error.
- Bayesian network is controlled by optimal updating coefficient through optimizing the estimation error.
- Software effort is estimated based on the number of defects in different steps of the software development thus cost and quality can be considered simultaneously.

We summarize the related models and their properties in Table 1.

In the section 2, we explain the proposed model without considering the software quality. In the section 3, we present the model with considering the quality in terms of the number of defects. In the section 4, we analyze the computational results of two models and finally, in the section 5, we express conclusion and future research opportunities.

2. Proposed model

In this section, we estimate the effort (man-month) without considering the number of defects. Flowchart of model is shown in Fig. 1.

The pseudo code of proposed model is presented in Tables 2 and 3. We divide the proposed model to four sections including building the Bayesian belief network based on COCOMO components, learning process, optimal control and test the model. In the first step, we estimated the effort for development of 40 projects based on Bayesian network. In second one, we compare the estimated values of efforts with actual values obtained from dataset. In optimal control section, we minimize the error measurement using particle swarm optimization and genetic algorithm and obtain optimal updating coefficients. In test section, we apply both coefficients to test the estimated effort of 20 projects that were not involved in learning process. Each coefficient with less value of error is the optimal updating coefficient of model and then, we can modify the estimations with multiplying the results with this coefficient.

[1] Delivered source instructions per man-month.

Table 1
Summary of related models to the software effort estimation.

Paper number	Year	Bayesian Network	The intervals of Bayesian Network are fuzzy	COCOMO Components	Considering SIZE	Estimation based on development steps	Estimation based on components	Comparing with datasets	Comparing with other methods	Optimal Control	Capability of estimation updating	Basic	Developed	Practical
[9]	1999	*		*	*		*	*	*		*			*
[13]	2003	*		*			*						*	
[14]	2004	*				*					*		*	
[10]	2005	*					*	*	*		*			*
[2]	2011	*			*		*	*		*	*		*	
[11]	2012	*			*		*	*			*			*
[3]	2013	*			*	*	*	*	*			*		
[1]	2013	*			*		*	*		*			*	
[15]	2013						*	*	*	*			*	
[17]	2014			*	*		*	*	*	*			*	
[4]	2014			*	*		*	*	*	*			*	
[18]	2015			*	*		*	*					*	
Proposed Method		*	*	*	*	*	*	*	*	*	*		*	

2.1. Bayesian network model based on COCOMO components

COCOMO NASA dataset[2] includes 63 implemented software projects which considers 15 COCOMO components, the software size based on 1000 lines of the code (KLOC) and the actual values of needed effort for software development. We use 40 projects for the model learning and other 20 projects are used for model testing. A part of the dataset is shown in Table 4 .

First, to build Bayesian network, we need to obtain intervals of software size and actual values of effort in dataset projects. The data of software size is shown in Table 5.

Bayesian belief network works with discrete intervals thus the accuracy of estimations reduces. Therefore, we resolve this issue by converting these discrete intervals to fuzzy ones to increase accuracy. There are different methods to convert crisp numbers to fuzzy ones [19–21]. Some methods of converting a crisp number to an asymmetric triangular fuzzy are as following:

$$First\quad Method:\ (\min(x_i),\ mean(x_i),\ \max(x_i)) \tag{1}$$

$$(\min(x_1),\ median(x_i),\ \max(x_i)) \tag{2}$$

$$(\mu-\sigma,\ \mu,\ \mu+\sigma) \tag{3}$$

In this paper, we apply the second method. Therefore, the median of 40 values of the software sizes is used as the second component of the fuzzy number. This method is suitable for small datasets. According to Table 3, 2.2, 50 and 423 are minimum, second quarter and maximum value of the software sizes of 40 projects in dataset applied in learning process. This fuzzy number is shown in Fig. 2.

In the other words, if the size is less than 50 KLOC then the fuzzy number will be computed from the Eq. (4) otherwise it will be computed from the Eq. (5). Also, we use defined intervals in Table 6 to discretize the actual values of effort.

$$\begin{cases} \dfrac{x-2.2}{50-2.2},\ Nominal \\ 1-\dfrac{x-2.2}{50-2.2},\ Low \\ 0,\ High \end{cases} \tag{4}$$

$$\begin{cases} \dfrac{423-x}{423-50},\ Nominal \\ 1-\dfrac{423-x}{423-50},\ High \\ 0,\ Low \end{cases} \tag{5}$$

After discretization, we build three-level Bayesian network used in the proposed model as shown in Fig. 3.

We obtained the prior probability distribution of the needed effort based on Fig. 3 as follows (Table 7).

$$\pi_0(product) = \Sigma_{RELY,DATA,CPLX}\, p(product|RELY, DATA, CPLX)\,\pi_0(RELY)\,\pi_0(DATA)\,\pi_0(CPLX) \tag{6}$$

$$\pi_0(computer) = \Sigma p(computer|TIME, STOR, VIRT, TURN)\,\pi_0(TIME)\,\pi_0(STOR)\,\pi_0(VIRT)\,\pi_0(TURN) \tag{7}$$

$$\pi_0(personnel) = \Sigma p(personnel|ACAP, AEXP, PCAP, VEXP, LEXP)\,\pi_0(ACAP)\,\pi_0(AEXP)\,\pi_0(PCAP)\,\pi_0(VEXP)\,\pi_0(LEXP) \tag{8}$$

[2] Promise.site.uottawa.ca/SERopository/Datasets/cocomonasa-v1.arff .

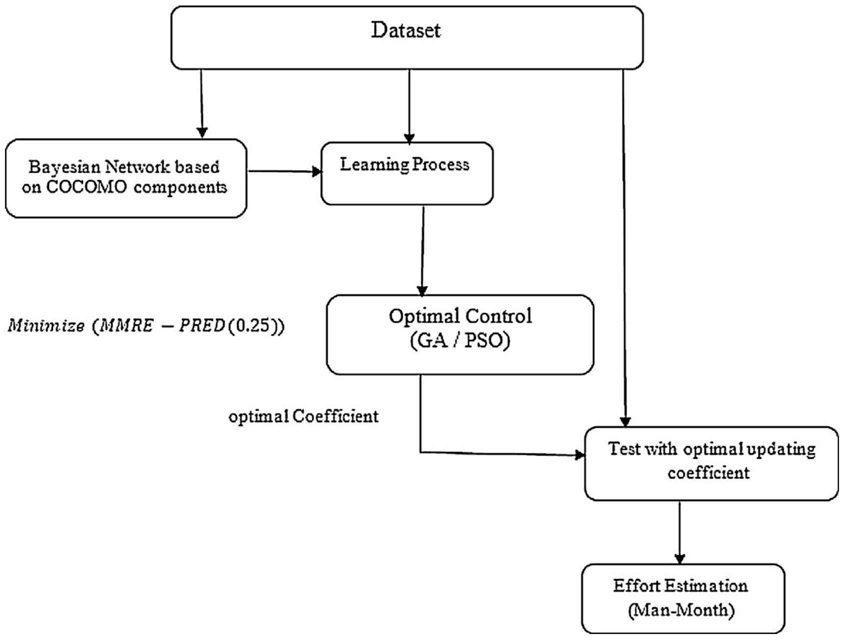
Dataset
Bayesian Network based on COCOMO components
Learning Process
Minimize (MMRE − PRED(0.25))
Optimal Control (GA / PSO)
optimal Coefficient
Test with optimal updating coefficient
Effort Estimation (Man-Month)

Fig. 1. Proposed model without considering the software quality.

Table 2
The pseudo code of proposed model.

```
%% BAYESIAN NETWORK MODEL BASED ON COCOMO COMPONENTS
for i=1:40
    % for each of 40 projects of learning process
    Calculate $\pi_{0i}$ based on **prob** function % inputs are described in the function
    Calculate *estimated effort* based on **BBN** function with the inputs: $\lambda_0, \pi_{0i}, mean$
end
%% LEARNING PROCESS
Calculate *Error* based on **LP** function with the inputs: *estimated effort, actual effort*
%% OPTIMAL CONTROL
    Calculate $\lambda_{GA}^{opt}$ based on **GA** function with the input: pop, cr, mr, ite
    Calculate $\lambda_{PSO}^{opt}$ based on **PSO** function with the inputs: $\lambda_0$
%% TEST THE MODEL WITH UPDATING COEFFICIENTS
for i=1:20
% for each of 20 projects of test process
    Calculate $\pi_0(effort_i)$ based on **prob** function *%inputs are described in the function*
    Calculate $estimated\ effort_i^{GA}$ based on **BBN** function with the inputs: $\lambda_{GA}^{opt}, \pi_{0_i}, mean$
    Calculate $estimated\ effort_i^{PSO}$ based on **BBN** function with the inputs: $\lambda_{PSO}^{opt}, \pi_{0i}, mean$
    Calculate $Error_{GA}$ Based on **LP** function with the inputs:
$estimated\ effort_i^{GA}, actual\ effort_i$
    Calculate $Error_{PSO}$ Based on **LP** function with the inputs:
$estimated\ effort_i^{PSO}, actual\ effort_i$
    if $Error_{GA} \le Error_{PSO}$ then
        $\lambda^{opt} = \lambda_{PSO}^{opt}$
        $estimated\ effort_i^{opt} = estimated\ effort_i^{PSO}$ %this estimation is the optimum estimation
        Display $estimated\ effort_i^{opt}$
    else
        $\lambda^{opt} = \lambda_{GA}^{opt}$
        $estimated\ effort_i^{opt} = estimated\ effort_i^{GA}$ %this estimation is the optimum estimation
        Display $estimated\ effort_i^{opt}$
    end if
end
```

$$\pi_0(project) = \Sigma p(project|MODP, TOOL, SCED)\pi_0(MODP)\pi_0(TOOL)\pi_0(SCED) \quad (9)$$

$$\pi_0(technicalfactors) = \Sigma p(technical\ factors|product, computer)\pi_0(product)\pi_0(computer) \quad (10)$$

$$\pi_0(humanfactors) = \Sigma p(human\ factors|personnel, project)\pi_0(personnel)\pi_0(project) \quad (11)$$

$$\pi_0(effort) = \Sigma p(effort|technical\ factors, human\ factors)\pi_0(technical\ factors)\pi_0(human\ factors) \quad (12)$$

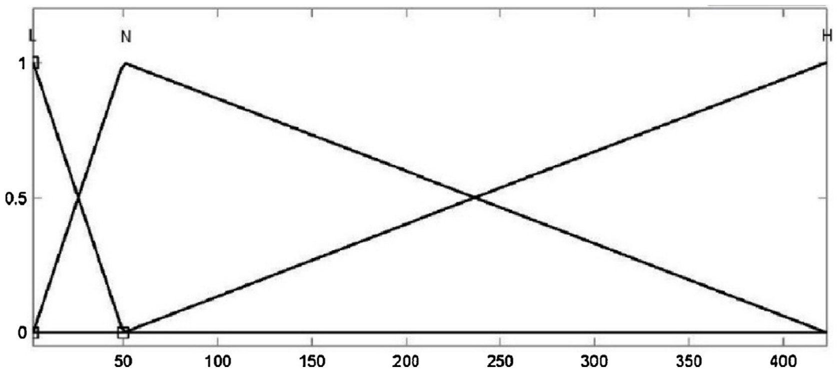

Fig. 2. conversion of the size parameter to a triangular fuzzy number.

Where $\pi_0()$ and $p(|)$ denote the prior probability distribution and the conditional probability, respectively. The conditional probabilities are obtained from dataset Tables 8–14 show the conditional probabilities of the Bayesian model. Then, the posterior probability distribution of the needed effort is determined as follows.

$$BEL\,(effort) = \alpha.\lambda_0\,(effort)\,.\pi_0\,(effort)\,,\,\alpha = [\lambda_0\,(effort)\,.\pi_0\,(effort)]^{-1} \quad (13)$$

Where $BEL\,(effort)$ denotes the posterior probability distribution of the needed effort or the same belief values for each of the three possible states of the needed effort. Also, $\lambda_0\,(effort)$ equals to$(1, 1, 1)$ in

Table 3
The functions of pseudo code of proposed model.

```
Functions:

Function name: BBN
Inputs: $\lambda$, $\pi_0$, $mean$
Output: $estimated\ effort_i$
Calculate $BEL(effort_i)$ based on formula 10
Calculate $estimated\ effort_i$ based on formula 11
Function name: prob
Inputs: $\pi_{0i}(RELY), \pi_{0i}(DATA), \pi_{0i}(CPLX), \pi_{0i}(TIME), \pi_{0i}(STOR), \pi_{0i}(VIRT), \pi_{0i}(TURN),$
$\pi_{0i}(ACAP), \pi_{0i}(AEXP), \pi_{0i}(PCAP), \pi_{0i}(VEXP), \pi_{0i}(LEXP), \pi_{0i}(MODP), \pi_{0i}(TOOL), \pi_{0i}(SCED)$
P(product| RELY, DATA, CPLX) % this conditional probability is described in table 6
P(computer| TIME, STOR, VIRT, TURN) % this conditional probability is described in table 7
P(personnel| ACAP, AEXP, PCAP, VEXP, LEXP) % this conditional probability is described in table 8
P(project| MODP, TOOL, SCED) % this conditional probability is described in table 9
P(technical factors| product, computer) % this conditional probability is described in table 10
P(human factors| personnel, project) % this conditional probability is described in table 11
P(effort| technical factors, human factors) % this conditional probability is described in table 9
Output: $\pi_{0i}(effort)$
Calculate $\pi_0(product)$ based on formula 6
Calculate $\pi_0(computer)$ based on formula 7
Calculate $\pi_0(personnel)$ based on formula 8
Calculate $\pi_0(project)$ based on formula 9
Calculate $\pi_0(technical\ factors)$ based on formula 10
Calculate $\pi_0(human\ factors)$ based on formula 11
Calculate $\pi_0(effort)$ based on formula 12

Function name: LP
Inputs: estimated effort, actual effort
Output: Error
Calculate $MRE_i$ based on formula 17
Calculate $MMER$ based on formula 16
Calculate $A$ based on formula 18
Calculate $Error$ based on formula 15

Function name: GA
Inputs: pop as the size of population
        cr as the crossover rate
        mr as the mutation rate
        ite as the iteration rate
Output: $\lambda_{GA}^{opt}$
 %% INITIALIZATION
      Generate pop feasible solutions randomly % each chromosome (x) has 12 bits (b)
      Save them in the population pop
%% LOOP UNTIL THE TERMINAL CONDITION
     For i=1:ite
         % for each iteration of algorithm
         %% CROSSOVER
              $nc = cr * pop$ % nc as the number of crossover
```

Table 3 (*Continued*)

```
            for j=1:$nc$
                Select randomly two solutions $x_A$ and $x_B$ from $pop$
                Generate $x_C$ and $x_D$ by one-point crossover to $x_A$ and $x_B$
                Save $x_C$ and $x_D$ to $pop_1$
            end
        %% MUTATION
            $nm = mr * pop * 12$ % $nm$ as the number of mutation
            for j=1:$nm$
                Select randomly a bit ($b$) from $pop_1$
                Mute that selected bit ($b$) and generate a new solution $\acute{x}_j$
                Update $x_j$ with $x'_j$ in $pop_1$
        %% UPDATING
            Update $pop = pop_1$
%% RETURNING THE BEST SOLUTION
    Calculate $\lambda = (\lambda^l, \lambda^n, \lambda^h)$ related to each $x$ form $pop_1$ based on 12 bits of $x$ and below relations:
    $\lambda^l = b_1.2^3 + b_2.2^2 + b_3.2^1 + b_4.2^0$
    $\lambda^n = b_5.2^3 + b_6.2^2 + b_7.2^1 + b_8.2^0$
    $\lambda^h = b_9.2^3 + b_{10}.2^2 + b_{11}.2^1 + b_{12}.2^0$
    Calculate the fitness value of all $x$ in $pop$ based on formula 19
    Set the best fitness value as $f^{best}$
    If $f^{best}$ is better than $f^{opt}$related to $\lambda_{GA}^{opt}$ in the history
    then
        Set $\lambda$ related to $f^{best}$ as the $\lambda_{GA}^{opt} = (\lambda_{GA}^{opt.l}, \lambda_{GA}^{opt.n}, \lambda_{GA}^{opt.h})$
    end if
    End
    Display $\lambda_{GA}^{opt}$

Function name: PSO
Inputs: $pop$ as the size of population
        $c_1$ as the cognition coefficient
        $c_2$ as the social coefficient
        $W$ inertia weight
        $ite$ as the iteration rate
Output: $\lambda_{PSO}^{opt}$
%% initialization
    Generate $pop$ feasible solutions randomly % a solution is $\lambda = (\lambda^l, \lambda^n, \lambda^h)$ in [0,1]
    Save them in the population $pop$
%% Loop until the terminal condition
    For j=1:$ite$
        % for each iteration of algorithm
        for i=1:$pop$
          % for each particle of population
          Calculate the fitness function based on formula 19
          If the fitness value is better than the best fitness value ($p_{best}$) in the history of that particle
          then
              Set current value as the new $p_{best}$
            end if
          If the fitness value is better than the best fitness value ($g_{best}$) in the history of population
          then
              Set the current value as the new $g_{best}$
           end if
        end
        for i=1:$pop$
          % for each particle of population
          Calculate particle velocity based on formula 24
          Apply the velocity constriction
          Update particle position based on formula 25
          Apply the position constriction
        end
      End
      Display $g_{best}$as the $\lambda_{PSO}^{opt} = (\lambda_{PSO}^{opt.l}, \lambda_{PSO}^{opt.n}, \lambda_{PSO}^{opt.h})$
```

Table 4
Scheme of COCOMO NASA dataset.

COCOMO NASA dataset																
		Human factors								Technical factors						
Real effort	Size	Project			Personnel					Computer				Product		
		SCED	TOOL	MODP	LEXP	VEXP	PCAP	AEXP	ACAP	TURN	VIRT	STOR	TIME	CPLX	DATA	RELY
170	32.6	L	VH	VH	H	L	N	H	VH	H	L	VH	VH	H	VH	N
324	150	N	N	N	H	N	VH	VH	H	L	L	XH	N	H	L	N
2400	302	N	VL	H	N	N	H	N	N	L	L	N	N	H	L	H
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
155	19.3	H	L	L	H	N	H	H	H	H	N	N	VH	H	H	N

VL: Very Low, L: Low, N: Nominal, H: High, VH: Very high, XH: Extra high.

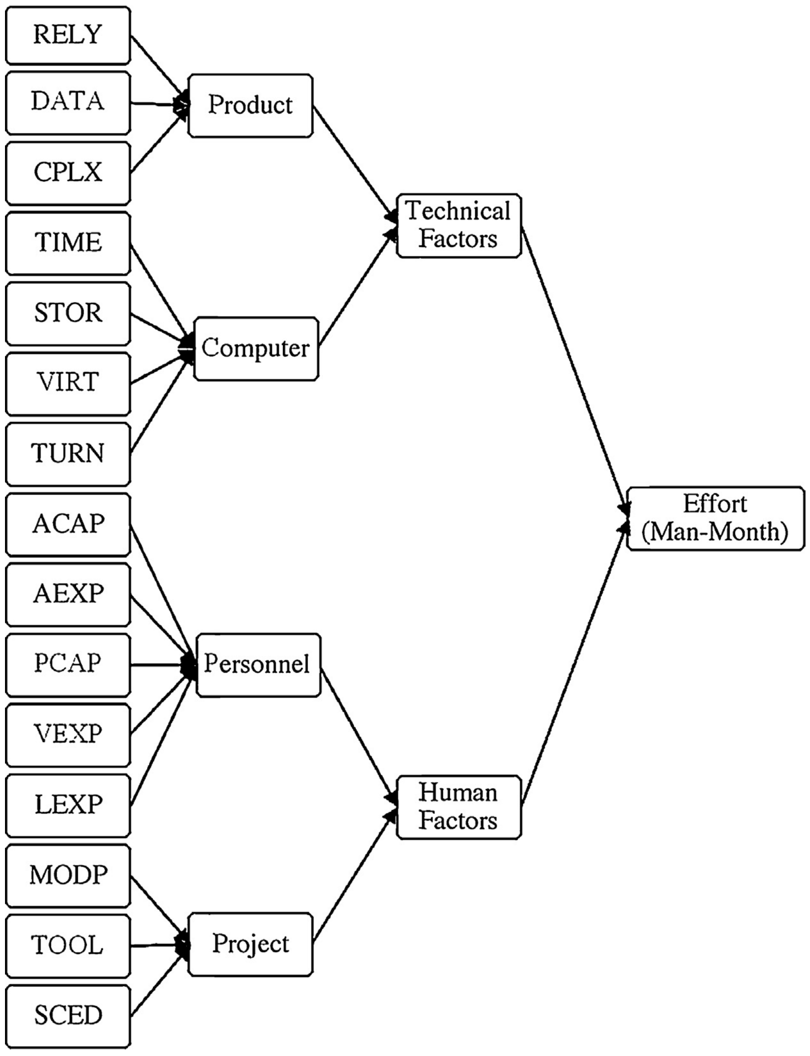

Fig. 3. Probabilistic three-level Bayesian network.

Table 5
Minimum and maximum value of the software sizes.

Software size	Minimum	Maximum
(KLOC)	2.2	423

Table 6
Defined intervals to discretize the actual values of efforts.

The level of real effort	Low	Nominal	High
The value of real effort (Man-Month)	[8.4–60]	[60–160]	[160–3240]
Mean	35.236	95.32	901.99

the first run of the procedure. Finally, the needed effort is estimated as follows.

$$est.effort = BEL(effort) * mean(effort) \quad (14)$$

Where *est.effort* denotes the estimated values of needed effort. *mean*(*effort*) equals to (35.236, 95.32, 901.99) based on the results of Table 6.

2.2. Learning approach

The prior probabilities of 40 projects applied in learning process of the model are computed where for each component related to the mentioned level at dataset, prior probability is equal to one and for the other levels, it equals to zero. For example, if CPLX component related to a certain project is “Low” then, $\pi_0(CPLX = Low) = 1$ and the values of π_0 related to the other levels equals to zero. Using prior probabilities and tables of conditional probabilities, the effort is estimated and then it is compared to the specified value of the actual effort and the error term is computed. In this model, the error term is computed by the Eqs. (15)–(18).

$$Error = Minimize(MMRE) - Maximize\left(\frac{A}{N}\right) \quad (15)$$

$$MMRE = \frac{\Sigma_{i=1}^{N} MRE_i}{N} \quad (16)$$

$$MRE_i = \frac{|est.effort_i - Act.effort_i|}{Act.effort_i} \quad (17)$$

$$A = \Sigma_{i=1}^{N} Y_i, \begin{cases} Y_i = 1, if, MRE_i \le 0.25 \\ Y_i = 0, if, MRE_i \ge 0.25 \end{cases} \quad (18)$$

MRE: magnitude relative error of effort estimation
MMRE: mean magnitude relative error of effort estimation
N: the number of projects
A: the number of projects that their *MRE* s are less than 0.25

Table 7
Notations and their definitions.

Notation	Definition
$\pi_0(CPLX)$	The prior probability distribution of CPLX; for example $\pi_0(CPLX = low) = 1, \pi_0(CPLX = otherwise) = 0$
$p(product\|RELY, DATA, CPLX)$	The conditional probability of product based on three components of RELY, DATA and CPLX
$\pi_0(product)$	The prior probability distribution of product
$\lambda_0(effort)$	The likelihood representing diagnostic support for the propositions $effort = low, effort = nominalandeffort = high$ [22]
α	A normalizing constant that ensures the total probability sums to unity [22]
$BEL(effort)$	The posterior probability distribution or the vector of belief values for each of the three possible states of *effort* [22]
$mean(effort)$	Mean of the actual effort belonged to three states of low, nominal and high and equals to (35.236, 95.32, 901.99)
est.effort	The estimated effort needed for software development
MMRE	Mean magnitude of relative error
MRE	Magnitude of relative error
Act.effort	The actual effort expended for software development
$\lambda^{opt}(effort)$	The optimal likelihood for the propositions $effort = low, effort = nominalandeffort = high$
$BEL^{opt}(effort)$	The optimal posterior probability distribution or the vector of belief values for each of the three possible states of *effort*

Table 8
Conditional probabilities of Product group based on three components.

RELY	*DATA*	*CPLX*	$p(product\|RELY, DATA, CPLX)$		
			$product =$ *Low*	$product =$ *Nominal*	$product =$ *High*
N	VH	H	0	0.8	0.2
H	L	H	0	0.86	0.14
N	L	H	0	0.95	0.05
H	N	H	0	0.81	0.19
H	N	VH	0	0.71	0.29
N	N	N	0	1	0
N	N	H	0	0.91	0.09
VH	N	XH	0	0.22	0.78
L	H	N	0.24	0.76	0
H	H	N	0	0.86	0.14
H	H	L	0	0.97	0.03
N	H	L	0.16	0.84	0
N	H	H	0	0.86	0.14

Table 9
Conditional probabilities of Computer group based on four components.

TIME	*STOR*	*VIRT*	*TURN*	$p(computer\|TIME, STOR, VIRT, TURN)$		
				$computer =$ *Low*	$computer =$ *Nominal*	$computer =$ *High*
VH	VH	L	H	0	0.84	0.16
N	N	L	L	1	0	0
N	XH	L	L	0	0.94	0.06
N	N	H	L	0	1	0
XH	XH	L	H	0	0.51	0.49
H	H	L	H	0	0.97	0.03
N	N	L	N	0.53	0.47	0
N	H	N	N	0	0.98	0.02
H	H	L	L	0.45	0.55	0
N	N	N	H	0	0.98	0.02
N	N	H	N	0	0.95	0.05
VH	N	N	H	0	0.86	0.14

Table 10
Conditional probabilities of Personnel based on five components.

LEXP	*VEXP*	*PCAP*	*AEXP*	*ACAP*	$p(personnel\|ACAP, AEXP, PCAP, VEXP, LEXP)$		
					$personnel =$ *Low*	$personnel =$ *Nominal*	$personnel =$ *High*
H	L	N	H	VH	0	0.5	0.5
H	N	N	N	N	0	0.92	0.08
H	N	VH	XH	H	0	0.18	0.082
VL	N	N	N	H	0	0.97	0.03
VL	L	H	H	H	0	0.76	0.24
H	N	H	VH	H	0	0.35	0.65
H	N	N	H	H	0	0.61	0.39
H	N	H	H	H	0	0.45	0.55
L	N	N	VH	H	0	0.62	0.38
N	N	H	N	N	0	0.78	0.22
N	N	N	H	N	0	0.86	0.14
H	H	N	N	H	0	0.59	0.41
H	L	VH	VH	H	0	0.26	0.74
H	N	VH	H	H	0	0.26	0.74
H	N	H	H	N	0	0.61	0.39
N	N	N	H	H	0	0.67	0.33
H	N	H	H	H	0	0.45	0.55

Table 11
Conditional probabilities of Project group based on three components.

SCED	*TOOL*	*MODP*	$p(project\|MODP, TOOL, SCED)$		
			$project =$ *Low*	$project =$ *Nominal*	$project =$ *High*
VH	VH	L	0	0.17	0.83
H	N	L	0	0.95	0.05
N	N	N	0	1	0
H	H	N	0	0.46	0.54
H	VL	N	0.05	0.95	0
L	VH	N	0	0.73	0.27
L	VH	H	0	0.84	0.16
H	N	N	0	0.72	0.28
H	N	H	0	0.83	0.17
L	H	N	0	1	0
L	L	H	0.1	0.9	0

Table 12
Conditional probabilities of Technical factors based on two components.

product	*computer*	$p(technicalF.\|product, computer)$		
		$technicalF. =$ *Low*	$technicalF. =$ *Nominal*	$technicalF. =$ *High*
N	N	0.01	0.92	0.07
N	L	0.19	0.805	0.005
N	H	0.01	0.92	0.07
H	N	0.01	0.92	0.07
L	N	0	0.99	0.01
L	H	0	0.99	0.01
H	H	0.01	0.92	0.07
H	L	0.172	0.823	0.005

Table 13
Conditional probabilities of Human factors based on two components.

project	*personnel*	$p(humanF.\|project, personnel)$		
		$humanF. =$ *Low*	$humanF. =$ *Nominal*	$humanF. =$ *High*
N	N	0	0.638	0.362
H	N	0	0.638	0.362
N	H	0	0.622	0.378
N	L	0	0.746	0.254

Table 14
Conditional probabilities of Effort based on three components.

technical	*human*	*size*	$p(effort\mid technical, human, size)$		
			effort = Low	*effort = Nominal*	*effort = High*
L	N	L	0.9	0.1	0
L	H	L	0.9	0.1	0
L	N	N	0.33	0.67	0
L	N	H	0	0	1
N	N	L	0.563	0.437	0
N	N	N	0.22	0.33	0.45
N	N	H	0	0	1
N	H	L	0.563	0.437	0
N	H	N	0.22	0.33	0.45
N	H	H	0	0	1
H	N	L	0.563	0.437	0
H	N	N	0.22	0.33	0.45
H	N	H	0	0	1
H	H	L	0.563	0.437	0
H	H	N	0.22	0.33	0.45
H	H	H	0	0	1
L	H	N	0.33	0.67	0
L	H	H	0	0	1

According to error measurement (formula 15), it's clear that this measurement can be positive or negative.

2.3. Optimal control

In this step, we consider the values of error obtained in the learning process as objective function and minimize it using genetic or particle swarm optimization algorithm. The optimal solution of the algorithm is indicated as $\lambda^{opt}(effort)$ which is used to modify the estimation determined by the Bayesian network. Modification method employed in optimal control is as following:

$$MinError = \frac{\Sigma_{i=1}^{40} MRE_i}{N} - \frac{A}{N} \tag{19}$$

$$A = \Sigma_{i=1}^{N} Y_i, \begin{cases} Y_i = 1, if, MRE_i \le 0.25 \\ Y_i = 0, if, MRE_i \ge 0.25 \end{cases} \tag{20}$$

$$MRE_i = \frac{|est.effort_i - Actual_i|}{Act.effort_i} \tag{21}$$

$$est.effort_i = BEL^{opt}(effort).mean(effort) \tag{22}$$

$$BEL^{opt}(effort) = \alpha.\lambda^{opt}(effort).BEL(effort) \tag{23}$$

2.3.1. Particle swarm optimization

This algorithm is inspired by the social behavior of birds and was suggested by Eberhart in 1995 [15]. In each iteration of this algorithm, the best local solution $p_{best}(i)$ and also, the best global one $g_{best}(i)$ are kept and used for computing the search velocity in next step. The solution of previous and current step are related to each other based on the velocity equation. The equations of velocity and location of particles are obtained in Eqs. (24)–(25).

$$v(i) = w * v(i-1) + c_1 * rand() * (p_{best}(i) - p(i)) + c_2 * rand() * (g_{best}(i) - p(i)) \tag{21}$$

$$p(i) = p(i-1) + v(i) \tag{22}$$

The optimal values of w, c_1 and c_2 are investigated in the reference [15]. If the value of w decreases then the algorithm goes more toward local search and when the value of w increases then the algorithm goes more toward global search. Here, λ^{opt} has the same role of $p^{opt}(i)$ which is the optimal solution of the algorithm.

2.3.2. Genetic algorithm

Genetic algorithm was presented by John Holand in 1975. This algorithm stems from the evaluation of the nature. In the other words, the consistent genes with the environment are remained and the weak genes are destroyed. Similarly, during the steps of solving a problem by this algorithm, bad solutions that are far from optimal solutions are deleted and the better ones are remained with higher probabilities. Thus, optimal or near optimal solution is obtained. During the run of this algorithm, operators of crossover and mutation and are implemented. In this paper, the solutions (λ) have three levels between 0 and 1. The pseudo code of GA is shown in Table 3. The applied genetic algorithm has following properties:

Generate initial population: We generate the initial population randomly. Each chromosome include 12 binary bits. Each bit is shown by b.

Chromosome representation: As shown in Fig. 4, the first four bits refer toλ^l (low level), the second four bits refer to λ^n (nominal level) and the third four bits refer to λ^h (high level). $\lambda = \left(\lambda^l, \lambda^n, \lambda^h\right)$ is the same as the updating coefficient that must be optimized by GA (λ_{GA}^{opt}). λ^l, λ^n and λ^h are computed as $\lambda^l = b_1.2^3 + b_2.2^2 + b_3.2^1 + b_4.2^0$, $\lambda^n = b_5.2^3 + b_6.2^2 + b_7.2^1 + b_8.2^0$ and$\lambda^h = b_9.2^3 + b_{10}.2^2 + b_{11}.2^1 + b_{12}.2^0$.

Fitness function: is $Error = \frac{\Sigma_{i=1}^{40} MRE_i}{N} - \frac{A}{N}$ or the same as Eq. (19)

Crossover operation: we use one-point crossover with the crossover rate = 0.6.

2.4. Testing the model

The first 20 projects of NASA dataset are considered for testing of the model. We compute the prior probability distributions of 15 components in first level of Bayesian network as mentioned in section 2.2 and obtain the prior probability distribution of needed effort using the tables of the conditional probabilities based on the Eqs. (6)–(12). The posterior probability distribution of the effort is computed based on the Eq. (13) so that $\lambda_0(effort)$ equals to $\lambda^{opt}(effort)$ which is obtained in the algorithm of optimal control. The new equation for evaluating posterior probability distribution of the needed effort is as following.

$$BEL(effort) = \alpha.\lambda^{opt}(effort).\pi_0(effort) \tag{26}$$

Then the posterior probability distribution of needed effort is multiplied with the mean values of needed effort. The effectiveness of optimal updating coefficient is determined by comparing the estimated effort with the actual value of effort and computing the error measurement.

3. The effort estimation with considering quality in different steps of software development

The estimation method mentioned in the section 3 is developed based on the components and the steps of software development haven't been considered. Also, the quality of developed software isn't considered in this estimation method. Two measurements of DIR[3] and DRM[4] are important factors used in this study to evaluate the software quality. DIR shows the rate of defect detection and DRM shows the rate of defect removal. Three important steps of software development include requirements specification, design and coding where some defects may appear in each of these three steps considering the components level. For example, if the level of CPLX component is XH (extra high) then the number of defects in the steps of requirements specification, design and coding will be

[3] Defect Introduction Rate.
[4] Defect Removal Model.

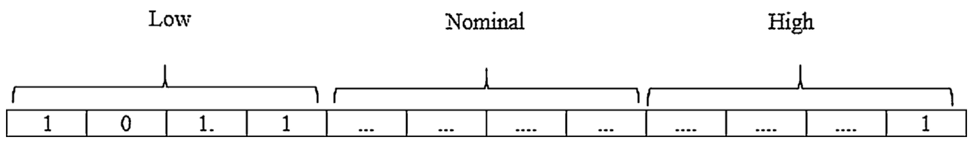

Fig. 4. Chromosome with 12 bits related to value of λ^{opt} ..

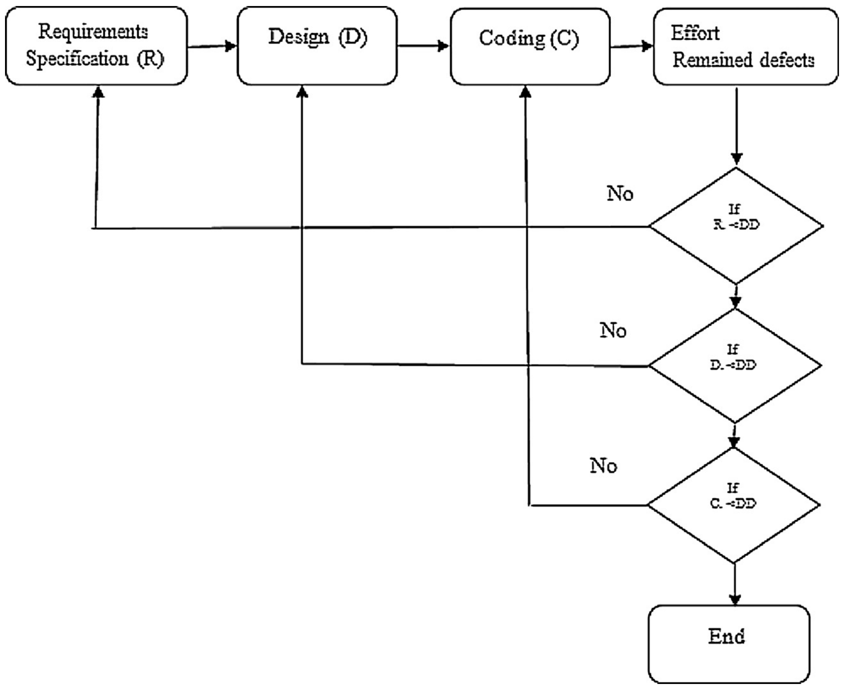

Fig. 5. The proposed model for software effort estimation with considering software quality in terms of the number of defects.

Table 15
DIR and DRM measurements for CPLX component.

CPLX level	DIR			DRM		
	R.	D.	C.	R.	D.	C.
XH	1.32	1.41	1.41	30%	25%	25%
N	1	1	1	50%	50%	50%
VL	0.76	0.71	0.71	75%	75%	75%
Final Quality Range	1.74	2	2	30–75%	25–75%	25–75%

Table 16
The baseline of DIR proposed in 1990.

Development Step	DIR
Requirements specification	9 per KLOC
Design	19 per KLOC
Coding	33 per KLOC

respectively 1.32, 1.41 and 1.41 times of the case that the level of component is N (nominal). Also, in this case 30, 25 and 25 percent of defects in the steps of requirements specification, design and coding are removed, respectively [16]. The values of DIR and DRM for CPLX component are shown in Table 15. Also, the values of other components are obtained in the reference [16]. The proposed model of the effort estimation with regards to these three steps and their remained defects is shown in Fig. 5. The baseline of DIR obtained in 1990 is shown in Table 16. The number of defects is converted to a fuzzy number using the triangular fuzzy distribution and the feedback loop is iterated based on the membership degree of “high” level. For example, if the membership degree of “high” level in the step of requirements specification is equal to 0.2 then, the final effort will be 1.2 times of estimated effort and the number of defects decreases based on mentioned percent in DRM tables.

Table 17
The optimal coefficient and error values obtained by genetic algorithm.

The number of iteration	The optimal error (GA)	λ^{opt}
50	0.264	(1,0,0.2)
250	0.282	(0.93,0.13,0.2)
500	0.298	(0.867,0.133,0.2)

4. The numerical analysis

In this section, we analyze the implementation of models in the cases of considering or not considering the number of software defects.

4.1. The numerical analysis of effort estimation without considering the quality in terms of the number of defects

4.1.1. The learning and optimal control

In the learning step, the estimation error without applying optimal coefficient was equal to 1.1764. The error term based on the optimal coefficient obtained by genetic algorithm is shown in Table 17. The genetic algorithm has following characteristics: crossover rate = 0.6, mutation rate = 0.01 and population with 6 chromosomes of 12 bits. Also, the obtained solution of PSO algorithm is shown in Table 18. The PSO characteristics includes population with 5 particles and maximum velocity between −0.5-0.5.

Table 18
The optimal coefficient and error values obtained by particle swarm optimization.

The number of iteration	The optimal error (PSO)	λ^{opt}
50	0.4631	(0.636,0.536,136)
250	**0.4626**	**(0.635,0.535,0.135)**
500	0.4631	(0.636,0.536,0.136)

The bold values show the optimal values.

Table 19
The solution of testing the model.

λ^{opt}	The error measurement	$A = \Sigma_{i=1}^{N} Y_i, \begin{cases} Y_i = 1, if, MRE_i \leq 0.25 \\ Y_i = 0, if, MRE_i \geq 0.25 \end{cases}$
(1,1,1)	0.47	2
(1,0,0.2)	**−0.082**	**10**
(0.635,0.535,0.135)	0.154	4

The bold values show the optimal values.

Table 20
Comparing the proposed model with the other models.

Models	Optimization method	λ^{opt}	The error measurement
1	PSO	(0.504,0.004,0.504)	1.30
2	PSO	(0.501,0.401,0.001)	0.341
3	PSO	(0.518,0.418,0.018)	0.303
4	GA	(1,0.467,0.067)	0.176
The proposed model	GA	(1,0,0.2)	−0.082

Table 21
The discrete intervals of size.

Size	Low	Nominal	High
(KLOC)	[2.2–13.9]	[13.9–100]	[100–423]

The results in Tables 17 and 18 show that in the proposed model and with mentioned conditions, the GA performs better than PSO. In the other words, the GA could decrease the error values from 1.1764 to 0.264.

4.1.2. Test results

The error value is equal to 0.47 based on 20 projects of NASA dataset when designed Bayesian network is implemented without considering the optimal coefficient. The error values obtained by the optimal coefficients determined using two algorithm and also, the number of projects with MRE values less than 0.25 are shown in Table 19.

The results in Table 19 show that the optimal updating coefficient (1, 0, 0.2) decreases the error from 0.47 to −0.082 and also, the number of projects with MRE values less than 0.25 increases from 2 projects to 10 ones.

4.1.3. Comparison with other models

We compare the results of the proposed model with four other models and the results are reported in Table 20.

Model 1

The reference [16] used the basic relation $E = a * size^b$ to estimate effort. According to this formula, we built the Bayesian network based on only size of software to estimate the effort (Fig. 6).

In this model the intervals of the software size (low, nominal and high) are discrete. The intervals of low, nominal and high are discretized between minimum and first quarter, the first quarter and third one and the third quarter and maximum value of the software sizes of 40 projects in NASA dataset. The intervals of the software size are shown in Table 21.

Model 2

Table 22
The number of remained defects after one time implementation of the model.

Project No.	Size	Requirements specification			Design			Coding		
		9*size	DIR/KLOC	DIR/KLOC *size	19*size	DIR/KLOC	DIR/KLOC *size	33*size	DIR/KLOC	DIR/KLOC *size
21	32.6	293.4	3.45	112.47	619.4	7.13	232.48	1075.8	11.62	378.68
26	24.6	233.1	4.07	105.52	492.1	8.37	216.85	811.8	14.21	349.62
34	150	1350	3.55	531.9	2850	7.27	1090.6	4950	12.39	1857.9
35	100	900	4.3	429.6	1900	9.02	901.87	3300	15.69	1568.6
...	...	...	...	...	...	...	...	...	...	...
Minimum DIR/KLOC			3.16	The fuzzy number (0,3.795,9)	7.13	The fuzzy number (0,8.29,19)	10.67	(0,13.44,33)		
Maximum DIR/KLOC			4.43		9.45	16.21				
Median DIR/KLOC			3.795		8.29	13.44				

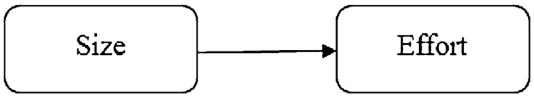

Fig. 6. The Bayesian belief network based on size of software project.

Table 23
The results of proposed model for the effort estimation with considering the development steps and the number of defects for the project 35 with size 100 KLOC.

DIR/KLOC and degree of membership of "high" level	Requirements specification	Design	Coding
DIR/KLOC after one time implementation of three steps	4.3	9.02	15.69
Degree of membership of "High" level	0.097	0.068	0.115
Updating DIR/KLOC after 1.097 iterations of loop of requirements specification	2.05	4.28	7.46
Degree of membership of "High" level	0	0	0

Final effort = 1.097*(effort estimation).

This model is the same as the model 1. However, the intervals of the software size are fuzzy numbers as depicted in Fig. 2. As reported in Table 20, the result of this model is better than model 1.

Model 3

This model is based on the reference [13]. A three-level Bayesian belief network is designed so that all intervals are discrete. The intervals of components including product, computer, personnel, technical and human factors, size and effort are assumed to be discrete. We control the model by PSO method.

Model 4

This model is the same as the model 3. However, we control this model by GA method.

The proposed model.

We proposed a three-level Bayesian belief network so that all intervals of components except effort are assumed to be fuzzy numbers that help us increase the accuracy of estimation as reported in Table 20.

4.2. The numerical analysis of effort estimation with considering the quality in terms of the number of defects

The baseline for DIR and also, the number of remained defects based on the software size after one time implementation of the model without applying feedback loop are presented in Table 22.

The second column is the software size based on 1000 lines of code. The third column shows the baseline of defects. Three numbers 9, 19 and 33 are shown in Table 16 that denote the baseline of DIR. We suppose that the process of software development in each step will have defects that the number of defects is equal to the number of baseline defects per kilo lines of code. Therefore, we consider 9*size as the potential defects in the requirements specification that will be reduced after implementing this step based on project condition, sources and customer needs. For example, the project number 35 with size 100 will have potential 900 defects in requirements specification step. After implementing the step, the number of defects reduces to 429.6 or 4.3 defects per KLOC. According to Table 23, 4.3 belongs to "high" level with membership value = 0.097. Therefore we should implement this step 1.097 times.

The results in Table 23 show that after one time implementation of development steps, the membership degree of "high" level that is related to the step of requirements specification is equal to 0.097. Also, after 1.097 iterations of feedback loop, the membership degree of "high" level for all steps reaches 0 and the desire level of software quality is achieved. It is important that COCOMO NASA dataset doesn't consider the number of defects. Thus, we can't compare the values of the final effort estimation with the actual values of the efforts in this dataset.

5. Conclusion

In this study we presented a three-level Bayesian network based on COCOMO components to estimate the needed effort (Man-Month) for the software development so that the estimated effort is modified using the optimal coefficient resulted from optimal control designed by the genetic algorithm. Also, we considered the intervals of network nodes except effort one as fuzzy numbers instead of discrete ones that increases the accuracy of estimation. We showed that the optimal coefficient resulted from the both GA and PSO can decrease the error values. However, the error value in GA method are less than the error values of PSO. Also, comparing the results of the proposed model with other models show that the accuracy of the proposed model is better than the results of other models.

Also, we considered the software quality in terms of the number of defects in the development steps and proposed a model to estimate the needed effort based on the number of defects so that if the number of defects in one step becomes more than defined thresholds then, that step is repeated and an additional effort is added to the estimated one. Because in NASA dataset, the quality of software isn't considered thus we couldn't compare the resulted estimations of the model with the actual values of the efforts.

Since Bayesian networks don't have any feedback loops thus as future opportunity study, we refer to a Bayesian network that can cover feedback loops in the model of software quality so that the return loops can somehow participate in learning process of the Bayesian network. Also, it's useful to present a model that can convert the output values of the estimated effort to an interval or fuzzy number in order to increase the flexibility.

Acknowledgment

The authors are grateful to the three anonymous referees for their valuable comments and constructive criticism.

References

[1] N. Akhtar, Perceptual evolution for software project cost estimation using ant colony system, Int. J. Comput. Appl. 81 (14) (2013) 23–30.
[2] J. Khan, Z.A. Shaikh, A.B. Nauman, Development of intelligent effort estimation model based on fuzzy logic using Bayesian networks, Softw. Eng. Bus. Contin. Educ. 257 (2011) 74–84.
[3] R. Fuentetaja, D. Borrajo, C.L. López, J. Ocón, Multi-step generation of bayesian networks models for software projects estimations, Int. J. Comput. Intell. Syst. 6 (5) (2013) 796–821.
[4] C.S. Yadav, R. Singh, Tuning of COCOMO II model parameters for estimating software development effort using GA for PROMISE project data set, Int. J. Comput. Appl. 90 (1) (2014) 37–43.
[5] C. Lopez-Martin, C. Isaza, A. Chavoya, Software development effort prediction of industrial projects applying a general regression neural network, Empir. Softw. Eng. 17 (2012) 738–756.
[6] A. Idri, F.A. Amazal, A. Abran, Analogy-based software development effort estimation: a systematic mapping and review, Inf. Softw. Technol. 58 (2015) 206–230.
[7] C. Lopez-Martin, Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects, Appl. Soft Comput. 27 (2015) 434–449.

[8] N. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, Making resource decisions for software projects, in: Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), United Kingdom, 2004, pp. 397–406.
[9] S. Chulani, B. Boehm, B. Steece, Bayesian analysis of empirical software engineering cost models, IEEE Trans. Softw. Eng. 25 (4) (1999) 573–583.
[10] P.C. Pendharkar, G.H. Subramanian, J.A. Rodger, A probabilistic model for predicting software development effort, IEEE Trans. Softw. Eng. 31 (7) (2005) 615–624.
[11] M. Fernández-Diego, J.-M. Torralba-Martínez, Discretization methods for NBC in effort estimation: an empirical comparison based on ISBSG projects, in: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Lund, 2012, 20–21 Sept., 2012, pp. 103–106.
[12] L. Uusitalo, Advantages and challenges of Bayesian networks in environmental modelling, Ecol. Model. 203 (3/4) (2007) 312–318.
[13] I. Stamelos, L. Angelis, P. Dimou, E. Sakellaris, on the use of Bayesian belief networks for the prediction of software productivity, Inf. Softw. Technol. 45 (1) (2003) 51–60.
[14] S. Bibi, I. Stamelos, Software process modeling with bayesian belief networks, in: Proceedings of the 10th International Software Metrics Symposium, Chicago USA (2004, 14–16 September), 2004.
[15] V.K. Bardsiri, D.N.A. Jawawi, S.Z.M. Hashim, E. Khatibi, A PSO-based model to increase the accuracy of software development effort estimation, Softw. Qual. J. 21 (3) (2013) 501–526.
[16] S. Devnani-Chulani, Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension, University of Southern California, Computer Science Department, California, 1997.
[17] F.S. Gharehchopogh, I. Maleki, S.R. Khaze, A novel particle swarm optimization approach for software effort estimation, Int. J. Acad. Res. Part A 6 (2) (2014) 69–76.
[18] A.K. Jakhar, K. Rajnish, An empirical approach for estimation of the software development effort, Int. J. Multimed. Ubiquitous Eng. 10 (2) (2015) 97–110.
[19] C.H. Chen, A.F. Li, Y.C. Lee, A fuzzy coherent rule mining algorithm, Appl. Soft Comput. 13 (7) (2013) 3422–3428.
[20] C.H. Chen, J.S. He, T.P. Hong, MOGA-based fuzzy data mining with taxonomy, Knowl. Based Syst. 54 (2013) 53–65.
[21] F. Zarebaghiabad, H. Khademizare, A three- stage algorithm for software cost and time estimation in fuzzy environment, Int. J. Ind. Eng. Prod. Res. 26 (3) (2015) 193–211.
[22] M.L. Krieg, A Tutorial on Bayesian Belief Networks, DSTO Electronics and Surveillance Research Laboratory, Edinburgh, 2001.