Volume 5, Issue 7, July 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Novel Technique for Adaptive Neuro Fuzzy Inference Model using Gaussian, Triangular and Trapezoidal Membership Functions

Vidisha Agrawal, Vishal Shrivastava
Department of Computer Science and Engineering,
ARYA College of Engineering and IT, Jaipur, Rajasthan, India

Abstract: Software effort estimation is a method to predict the effort required to develop software project. This paper presents a general overview of software estimation models and techniques. Models can be categorized as Size-Based, Function-Based, Learning-Based and Expertise-Based. Both Size-based and Function-based models can be termed as Parametric as they use a function or formula of fixed form for software cost/effort estimation. Each and Every model has its own strengths and weaknesses. The key factor in choosing an estimation model is the accuracy of its estimates. Unfortunately, there is no single technique that is best for every situation, and that a careful comparison of the results of several approaches is most likely to produce realistic estimates. Basically in this paper presents an Adaptive Neuro-Fuzzy Approach for Software Development Time Estimation. This proposed technique is aimed at building and evaluating a Neuro - fuzzy model using three (3) membership functions (MFs) for software project development time. The forty one modules were used as a data set. Our proposed approach for Neuro fuzzy using 3 membership functions i.e. Gaussian MF (GMF), Triangular MF (Tri MF) and Trapezoidal MF (Trap MF) is compared with neural network models and the results show that values of various relative error parameters for Neuro-fuzzy is lower than the values of parameters applying neural network.

Keywords —Software development Effort Estimation, GMF, Tri MF and Trap MF,Neuro-fuzzy model.

I. INTRODUCTION

To develop a project successfully, it is important for any organization that the project should be completed within budget, on time and the project should have requisite quality. In order to create a successful project, cost estimation is essential in managing software projects because of the uncertainty and diversity nature intrinsic in project development. Estimation is the intelligent anticipation of quantum of the work that needs to be performed and the resources required to perform the work in a defined environment using specific methods [1]. Software cost can be defined as cost incurred in various resources to develop a software project. The most important resource to develop software is man power. Software estimation gives the approximate calculation of software size, software development cost and effort, and development schedule for a particular software project. Software development effort can be typify as the required human resources necessary for developing the software project of an estimated size. It is measured in "person-month" or "person-hours". Software development effort estimates are the basis for project bidding, budgeting and planning. It is all about the future prediction of the work so that the managers can make decisions that how long and how many resources are required to complete the project. The use of erroneous estimation makes the manager's decision as a recipe of disaster and looses the control and execute plan in a wrong direction.
Software estimation involves the determination of one or more of the following parameters:

- Effort (usually in person-months)
- Project duration (in calendar time)
- Cost (in dollars)

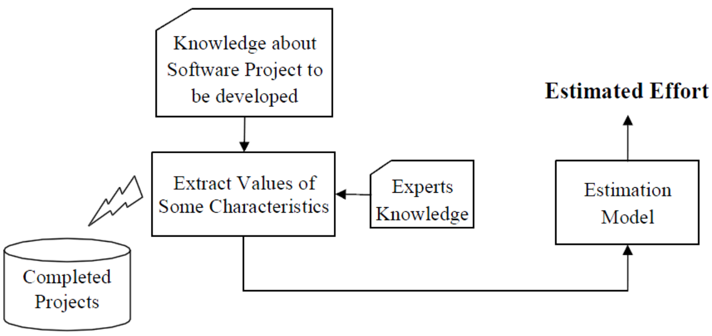

Fig 1: Software effort estimation process

In this paper, we present a fuzzy logic (FL) framework for effort prediction. The paper is organized as follows. In Section 2, we discuss the eventually development of both algorithmic and non-algorithmic models; Section 3 presents our soft computing-based prediction systems. Section 4 concludes discussions of our various experiments to realize the framework and points out potential directions for the upcoming research.

II. APPROACHES FOR SOFTWARE EFFORT ESTIMATION

There are number of estimation methods that have been developed, from very simple and earliest expert judgment to the more complex algorithmic modeling, analogy based methods and some hybrid techniques with soft computing. The vagueness of the expert judgment makes the estimation a critical task and motivates the researchers to develop more efficient methods for effort estimation [2][3]. There are several approaches for software cost estimation described in following section [4].

A. Algorithmic Models

From the study of historical data, costs are analyzed using mathematical formulae linking costs or inputs with metrics to produce an estimated output. These metrics are generally characterized of the target system and the implementation environment called cost drivers. Different algorithmic forms; Linear models (Nelson model, 1966), Multiplicative models (Doty model [Herd and others, 1977] and WalstonFelix,1977),

B. Expert Judgment

Expert judgment techniques involve consulting with software cost estimation expert or a group of experts to use their experience and understanding of the proposed project to reach at an estimate of its cost. This method is often used when estimating the effort needed to change an existing piece of software. One of the most common methods which work according to this technique is Delphi [3]. Delphi arranges a special meeting among the project experts and tries to attain the true information about the project from their debates.

The problems with this method are also associated:

- This method cannot be quantified.
- It is hard to document the factors used by the experts or experts-group.
- Expert may be some biased, optimistic, and pessimistic, even though they have been decreased by the group consensus.
- The expert judgment method always compliments the other cost estimating methods such as algorithmic method.

C. Analogy Based

Analogy is defined as “Inference that if two or more things agree with one another in some respects, they will probably agree in others”. In software cost estimation approach a similar completed project is identified and its actual effort is used as the basis of the estimate for the new project. It is infrequently used at the early stages of software development because of such inherent uncertainty and imprecision associated with attribute measurement. Analogy-based reasoning is often used, however, especially in software effort estimation as a synonym for case based reasoning (CBR), to describe the typical case-based approach where experience is retained for future reference.

Analogy follows the general case-based reasoning (CBR) process. This section provides a general overview of this process. Aamodt and Plaza describes a 4-stage general CBR cycle, which consists of [6]:

1. *RETRIEVE:*The most similar cases or cases to the target problem.
2. *REUSE:*The past information and solution to solve the new problem.
3. *REVISE:*The proposed solution and to better adapt the target problem.
4. *RETAIN*:The parts of current experience in the case-base for future problem solving.

The main advantages of this method are:

- The estimation is based on actual project characteristic data.
- The estimator's past experience and knowledge can be used which is not easy to be quantified.
- The differences between the completed and the proposed project can be identified and impacts estimated.

The disadvantages include:

- We have to determine how best to describe the projects. Which attributes are used having different influence on software effort?
- We have to determine the similarity and how much confidence can we place in the analogies. Uncertainty inherited in software projects makes it complex.
- Finally, we have to obtain an estimate for the new project by using known effort values from the analogous projects.

D. Parkinson Estimation

Parkinson’s Law [Parkinson, 1957] says, "Work expands to fill the available volume". In some cases, a Parkinson estimate has turned out to be remarkably accurate. These have generally been cases in which the estimate left a good deal of extra time and money to continue adding marginally useful "bells and whistles" to the software until the budget ran out at which point the software was declared complete. Parkinson estimation is not recommended. Besides not being particularly accurate, it tends to reinforce poor software development practice.

E. Price to Win

In this approach a figure that appears to be sufficiently low to win a contract is considered as 'estimate' [7]. The price-to-win technique has won a large number of software contracts for a large number of software companies. The inevitable result is that the money or schedule runs out before the job is done, everybody gets mad at each other, a lot of compromises are made about the software to be delivered, and a lot of programmers work long hours just trying to keep the Job from becoming a complete disaster.

F. Top-Down

Top-down approach is normally associated with parametric model. Top-down estimating method is also called Macro Model. Using top-down estimating method, an overall cost estimation for the project is derived from the global properties of the software project and then the project is partitioned into various low-level components. The leading method using this approach is Putnam model. This method is more applicable to early cost estimation when only global properties are known. In the early phase of the software development, it is very useful because there is no detailed information available [8].

The advantages of this method are:

- Does not need detailed information, so can be applied in early stages.
- All the estimates are done at system level that focuses on system-level activities such as integration, documentation, configuration management, etc.
- It is usually faster and easier to implement.

The disadvantages are:

- A revision of estimates makes large changes in schedule and time as each iteration gives more detail.
- It often does not identify difficult low-level problems that are likely to raise costs and sometime tends to overlook low-level components.
- It provides no detailed basis for justifying decisions or estimates.

G. Bottom-Up

It is also an important method of cost estimation process. Bottom-up estimation involves breaking the project into its component tasks and then estimates how much effort will be required to carry out each task, then combining the results to generate an estimate of the complete project. It is often difficult to execute a bottom-up estimate early in the life cycle process because the necessary information may not be available. This method also tends to be more time consuming and may not be practicable when either time or personnel are limited [9],

The advantages of this model include:

- It can be applied for completely novel project that has no historical data.
- It permits the software group to handle an estimate in an almost traditional fashion and to handle estimate components for which the group has a feel.
- It is more stable because the estimation errors in the various components have a chance to balance out.

The disadvantages are:

- It makes some assumptions about the characteristics of the final system because the necessary information may not available in the early phase.
- It may overlook many of the system-level costs (integration, configuration management, quality assurance, etc.) associated with software development.
- It tends to be more time-consuming.
- It may not be feasible when either time or personnel are limited.

III. COMPUTATIONAL INTELLIGENCE TECHNIQUES

Computational intelligence is the study of adaptive mechanisms to allow or facilitate intelligent behavior in complex and changing environments. As such, computational intelligence combines artificial neural networks, evolutionary computing, swarm intelligence and fuzzy systems. Software cost estimation systems are large complex nonlinear stochastic systems. Therefore, it is hard to find optimal feature weighting and project selection in any cost estimation model. Computational Intelligence provides a possible way to obtain either optimal or suboptimal solutions. Computational Intelligence methodologies can be adapted to dynamic changes in project parameters. Software cost estimation actions can be taken based on real-time datasets and historical reasoning. Researchers have conducted a lot of work for applications of computational intelligence in the field of software cost estimation [10], [11].

A. Evolutionary Computation (EC)

EC are nature inspired techniques. EC are essentially an umbrella of techniques that include genetic algorithms (GAs), genetic programming, evolutionary programming, evolutionary strategies, differential evolution, and so on. They reproduce natural processes, such as natural evolution under the principle of survival of the fittest. Fitness of a population indicates quality of solution that it represents. Out of the many performance metrics, MMRE is the de facto standard for software cost estimation that is mostly used as a fitness function.

B. Artificial Neural Network (ANN)

The feed forward multi-layer network with back propagation learning is the most commonly used structure in the field of software cost estimation. The network contains neurons arranged in layers with each neuron is connected to every neuron of the lower layer forming a complete graph. The cost drivers or project attributes are fed as inputs at the input layer which propagates across subsequent layers of processing elements known as neurons and generates effort estimation in terms of Person-Months (PM) at the output layer. ANN follows a two-step process. In step 1 threefold validation is employed for the training of the non-linear adjustment (ANN). This is followed by predicting stage in step 2. At this stage, a new project is presented to the trained system. The training process of an ANN is a non-linear and non-constrained optimization problem, where a search takes place for a minimum of the error function between the network output and the desired output. This cost function traditionally is the mean square error (MSE).

C. Fuzzy Logic

The three main steps to apply fuzzy logic for effort prediction are:

Step 1: Fuzzification: It converts crisp input to fuzzy output.

Step 2: Fuzzy Rule Based System: Fuzzy logic systems use fuzzy IFTHEN rules. Once all crisp input values are fuzzified into their respective Linguistic values, the fuzzy inference engine accesses the fuzzy rule base to derive.

Step 3: Defuzzification: It converts fuzzy output into crisp output.

An adaptive software effort estimation model incorporating different fuzzy logic system is developed to handle imprecision and uncertainty in software attributes of COCOMO-II model. Ahmed's Type-2 Fuzzy logic System (FLS) which evaluates the performance of a prediction system developed using the framework for handling imprecision and uncertainty when size is provided as a precise but uncertain input is an another example of fuzzy system software cost estimation. The prediction system consists of two stages: nominal effort prediction and EAF (Effort Adjustment Factor) prediction. The outputs of both the stages are merged (multiplied) to produce the actual effort.

IV. RESEARCH METHODOLOGY

The main goal of this paper is to evaluate software development time using an adaptive Neuro fuzzy approach. In this paper an Adaptive Neuro Fuzzy Inference System (ANFIS) tool is used. The network is trained by using learning algorithm i.e. Hybrid Approach (combination of back propagation and least mean square algorithm). This methodology consists of four steps: 1) Loading of Training Data and 2) Generating Fuzzy Inference System 3) Training of ANFIS 4) Development Time Estimation.

START

Step I; Determine the inputs of the model;

Step II; Generate ANFIS model;

Step III; Evaluate the value of Development Time;

Step IV; Evaluate the Value of MRE from result obtained by step III;

Step V; Evaluate the Value of MMRE and PRED from result obtained by step IV;

Step VI; Evaluate the Value of BRE from result obtained by step III;

END

4.1 Performance Evaluation Metrics

The following evaluation metrics are adapted to assess and evaluate the performance of the time estimation models.

(1) Magnitude of Relative Error (MRE)

$$MRE = \frac{|Actual\ \ Time - Estimated\ \ Time\,|}{Actual\ \ Time} \times 100 \qquad \text{Eq.(1)}$$

(2) Mean Magnitude of Relative Error (MMRE)

$$MMRE = \frac{1}{n}\sum_{\mathrm{i=1}}^{\mathrm{i=n}} \frac{|\mathrm{Actual\ \ Time - Predicted\ \ Time}\,|}{\mathrm{Actual\ \ Time}} \qquad \text{Eq.(2)}$$

The MMRE calculates the mean for the sum of the MRE of n projects. Specifically, it is used to evaluate the prediction performance of an estimation model.

(3) Prediction Level (PRED)

$$PRED(l) = \frac{k}{n} \times 100 \qquad \text{Eq.(3)}$$

where l is the maximum MRE of a selected range, n is the total number of projects, and k is number of projectsin a set of n projects whose MRE <= l. PRED calculates the ratio of projects' MREs that falls into the selected range (l) out of the total projects. (e.g. n = 100, k =80, where L= MRE <= 30%: PRED(30%)=80/100=80%).

(4) Balanced Relative Error (BRE)

$$BRE(\%) = \frac{|EstimatedTime\ \ - ActualTime\ \,|}{\min(T,T')} \times 100 \qquad \text{Eq.(4)}$$

Where T= estimated time and T'=actual time

V. RESULTS ANALYSIS AND DISCUSSION

Testing was carried out on a system with Intel ® core (i3), 2.93 GHz with 2GB RAM and implemented using MATLAB (7.10). For this experiment Lopez Martin dataset was used with 41 modules. Lopez Martin dataset is divided into two sets: Training dataset and testing dataset. Training dataset includes 25 modules and testing dataset includes 11 modules. Training dataset was chosen randomly from 41 modules with which we can achieve the best results. The three different membership functions were trained using the same 25 modules. We have divided the entire dataset into two sets, training set and testing set in the ratio of 100% and 20%. Training set consists of data from 25 projects selected randomly and testing set consists of 11 project samples that were used later on for testing. Evaluation of Neuro Fuzzy Model for Gaussian Membership Function.

The loading of the training data is done which consists of 25 modules as training data. Once the training data is loaded then we have to generate the Fuzzy Inference System for particular membership function. We are generating Fuzzy Inference file for Gaussian MF which consists of three MFs as Inputs each with different no. of membership functions. The type of MF is for Output is Constant by default for Adaptive Neuro Fuzzy. The below Figure 6 shows the ANFIS Model for Gaussian MF with 3 MF for McCabe Complexity, 3 MF for Dhama Coupling and 4 MF for LOC. The below figure consists of 3 Inputs and 1 Output Development Time. The no. of rules generated is 36 for the given no. of input membership functions.

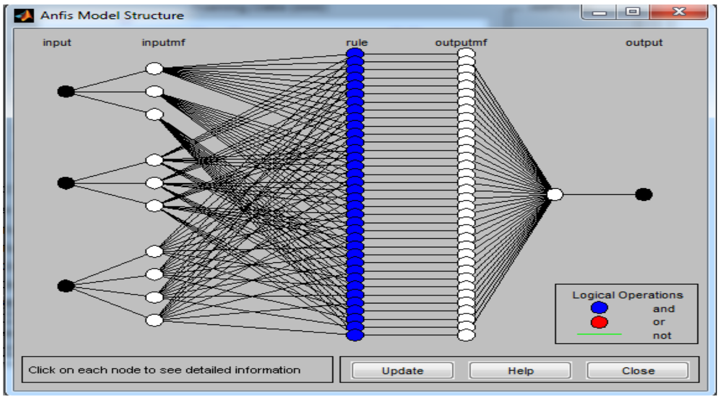

Figure 2: ANFIS Model Structure for Gaussian Membership Function

Graphical User Interface: An Interface is designed for representing our thesis work. Below are the various snapshots for our GUI.

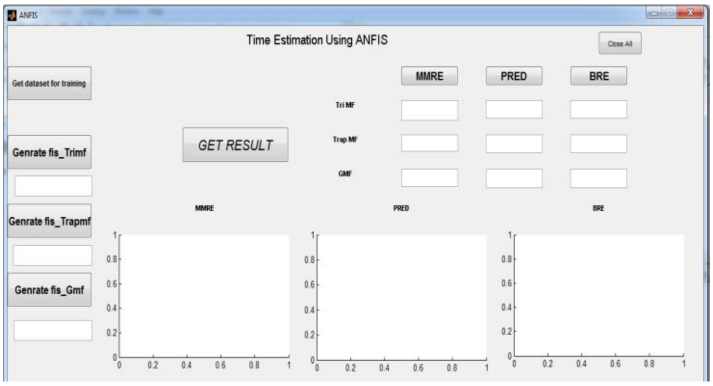

Figure 3: Graphical User Interface

The below figure 3 shows when the training data is loaded into the GUI. Once the training data is loaded then the fuzzy inferences files will be generated. Firstly, the Triangular Membership Function file is generated. Once this file is generated it is displayed fis file generated.

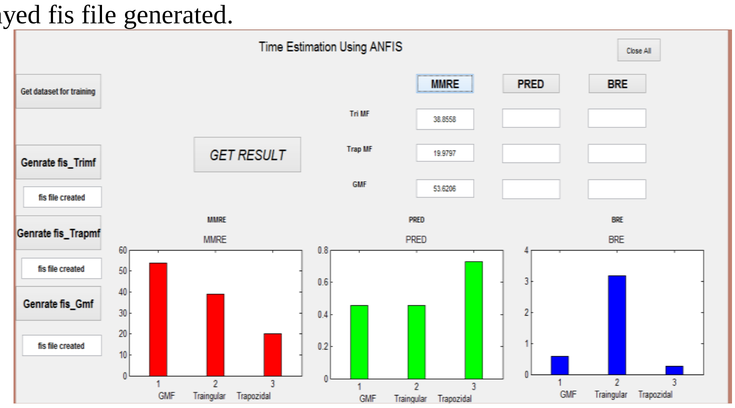

Figure 4: MMRE Results

The above figure 4 shows the results for MMRE for three membership functions i.e. Triangular MF, Trapezoidal MF and Gaussian MF.

The below figure 5 shows the results for PRED for three membership functions i.e. Triangular MF, Trapezoidal MF and Gaussian MF.

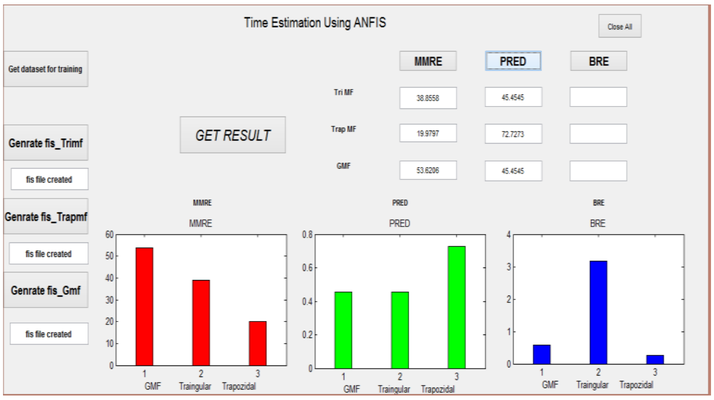

Figure 5: PRED Results

Figure 6: BRE Results

The above figure 6 shows the results for BRE for three membership functions i.e. Triangular MF, Trapezoidal MF and Gaussian MF.

VI. CONCLUSIONS

As none of the methods are satisfactory enough to fit in all circumstances which are frequent irrespective of Environments; it necessitates expertise as well as revelation to combine various techniques if possible and then calibrate. Differences between the estimated efforts can then be reconciled using statistical analysis techniques [14], [15].Applicability of using Soft Computing and Machine Learning Techniques to solve the effort and cost estimation problem for software systems. Use of artificial neural networks (ANNs), Genetic Algorithms (GAs), Genetic Programming (GP), Linear Regression (LR) and Fuzzy-Logic to present a methodology for software cost estimation. The paper suggests a new approach for estimating of software project development time. In this paper, Adaptive Neuro Fuzzy Inference model is considered and three membership functions i.e. Gaussian MF, Triangular MF and Trapezoidal MF are used to predict the future values in the case of graphical user interface. It is observed that Neuro Fuzzy model using Trapezoidal membership function gives better results than all other models. It is also observed that Trapezoidal MF gives better results for all the three parameters. In order to achieve more accurate estimation, the estimated values of several other techniques and combine their results may be useful.

REFERENCES

[1] M. Chemuturi, "Software Estimation Best Practices, Tools & Techniques: AComplete Guide for Software ProjectsEstimator",availableat:http://books.google.co.in/books?id=IwEOB2Mfzx0C&pg=PA1&source=gbs_toc_r &cad=4#v=onepage&q&f=false, 2009.

[2] B. W. Boehm, "Software Engineering Economics", Prentice-Hall, EnglewoodCliffs, NJ, USA, 1981.

[3] T. Gruschke, "Empirical Studies of Software Cost Estimation: Training of EffortEstimation Uncertainty Assessment Skills", 11th IEEE International SoftwareMetrics Symposium, IEEE, 2005.

[4] C. C. Kung and J. Y. Su, "Affine Takagi-Sugeno fuzzy modeling algorithm byFuzzy c-regression models clustering with a novel cluster validity criterion", IETControl Theory Appl., pp. 1255 – 1265, 2007.

[5] V. Khatibi, Dayang and N. A. Jawawi, " Software Cost Estimation Methods: AReview", Journal of Emerging Trends in Computing and Information Sciences,CIS Journal, Vol. 2, no. 1, ISSN 2079-8407, 2011.

[6] N. Sharma1, A. Bajpai and M. R. Litoriya, "A Comparison of Software CostEstimation Methods: A Survey", The International Journal of Computer Scienceand Applications (TIJCSA), Vol.1, no. 3, ISSN – 2278 – 1080, May 2012.

[7] J. Keung, "Software Development Cost Estimation Using Analogy: A Review",Australian Software Engineering conference, IEEE, 2009, DOI:10.1109/ASWEC.2009.32, 1530-0803/09.

[8] B. Hughes and M. Cotterell, "Software Project Management", Tata McGraw-Hill, 2006.

[9] N. Sharma1, A. Bajpai and M. R. Litoriya, “A Comparison of Software CostEstimation Methods: A Survey”, The International Journal of Computer Scienceand Applications (TIJCSA), Vol.1, no. 3, ISSN – 2278 – 1080, May 2012.

[10] T. R. Benala, S. Dehuri and R. Mall, “Computational Intelligence in Software CostEstimation: An Emerging Paradigm”, ACM SIGSOFT Software Engineering NotesPage, Vol. 37, no.3, 2012, DOI: 10.1145/180921.2180932.

[11] J.S. Pahariya, V. Ravi and M. Carr, “Software Cost Estimation using ComputationalIntelligence Techniques”, World Congress on Nature & Biologically InspiredComputing(NaBIC 2009)978-1-4244-5612-3/09/2009 IEEE, 2009.

[12] Qureshi, M. R. J. and S. A. Hussain. A Reusable Software Component-Based Development Process Model Int. J Advances in Engineering Software, 39(2): 88-94 (2008).

[13] Software Engineering – Kassem A. Saleh, J .Ross Publishinh, 2009.

[14] Software Engineering (Principles and Practice) – Waman S Jawadekar, TMH, 2004.

[15] Roger, S. P. Software Engineering: A Practitioner’s Approach. pp 722-.742 5th edi. McGraw-Hill (2000).

[16] Mrinal Kanti Ghose, Roheet Bhatnagar and Vandana Bhattacharjee. “Comparing Some Neural Network Models forSoftware Development Effort Prediction”, IEEE 2011.

[17] H. Mittal and P. Bhatia, “Optimization criteria for effort estimation using fuzzy technique,” CLEI ELECTRONIC JOURNAL, vol. 10, no. 1, pp. 1–11, 2007.

