International Journal of Computer Applications Technology and Research
Volume 5– Issue 1, 01 - 07, 2016, ISSN:- 2319–8656

Evaluating Reliability of Enterprise Architecture Based On Formal Fuzzy Models

S. Navaezadeh
Sama Technical and Vocational Training College,
Islamic Azad University, Mahshahr, Branch
Mahshahr, Iran

Rokhsareh Kabiri
Sama Technical and Vocational Training College,
Islamic Azad University, Mahshahr, Branch
Mahshahr, Iran

Abstract: The process of enterprise architecture (EA) is divided into strategic IT planning, EA development and EA implementation phases, respectively. Each phase is a prerequisite to the following phase. If the enterprise architecture planning (EAP) is not properly formulated during this process, its implementation would face problems and consequently a large amount of time and money would be wasted. EAP should be redefined to solve this issue. Definition of a method for the evaluation of EAP before the implementation of EA can be very useful in avoiding money and time loss caused by inappropriate EAP. A variety of methods are proposed around the world for evaluating EA but none of them can define and evaluate it in enterprises with uncertain processes or data. This research aims to present a new method for evaluating the reliability of EA while processes or data are uncertain. To achieve this goal, the products of fuzzy enterprise architecture (FEA) which demonstrate the behavior and sequence of processes and details of the enterprise are converted into fuzzy Petri nets. Afterwards, reliability evaluation is performed with fuzzy Petri nets which are executable models. In this method, uncertain processes and events can be easily modeled and evaluated. Therefore, the model proposed in this research is considered as a new method in EA. This method is executable and can be used in EA three-stage process.

Keywords: Enterprise Architecture Evaluation, Fuzzy Enterprise Architecture, C4ISR Framework, Fuzzy Colored Petri Nets

1. INTRODUCTION

The necessity of enterprise architecture can be observed in emerging large organizations, developing and designing complex information systems, emerging information systems for special purposes, the importance of organizations flexibility against external pressures like business changes, mission changes, enterprise structures and quick changes of technology. Organization architecture is a comprehensive method to describe future or current behavior and structure of organization processes, information systems and organization subunits. Therefore, organization architecture is presented and set on the basis of the main purpose of organization strategies. The main purpose of enterprise or organization architecture is to remove information technology as a tool, and to change it as organization resources along with other resources (financial, human, knowledge, experience and etc). They provide services for organization missions, and they should provide their own costs [3]. Before executing enterprise architecture, it's necessary to evaluate enterprise architecture so that the errors of modeling process are removed before final implementation. There are various frameworks for enterprise architecture modeling. In this paper, CaIsr

framework is used. CaisR framework is abbreviation of the following names: command, control, computers, communications, intelligence, supervision and recognition. It is a comprehensive framework and it describes the architecture with some determined document and with the name of product. Nowadays, no framework has implemented samples like CaIsR [1]. In today's world, process and events are uncertain and indecisive. Therefore, using fuzzy concepts in enterprise architecture guarantees that the model is closer to the real world.

Recently, due to importance and application of fuzzy concept, they are used in most modeling branches so that modeled system is closer to the real world. In this paper, a model of organization architecture is used, and it describes uncertainly and in decisiveness a processes and data of enterprise architecture by using fuzzy concepts. With regard to the fact that Petri nets are simple and are strongly supported, these networks can be used to create on executable model of enterprise architecture. Then, by using executable model, the behavior and requirements of non-functional organization architecture can be evaluated. Operational event trace description (OV-6C) is used in enterprise architecture to describe the sequence and schedule of operation and to trace the activities of a scenario and critical sequences [4].this product is described by a fuzzy format. In this paper, this product is converted to colored petri networks, and then reliability of enterprise architecture is evaluated by using colored fuzzy Petri network. Up to now, some methods have been presented to evaluate reliability by using formal networks like queue network, Petr. network, process algebra, Atamata and etc. In this paper, since synchronization of various simultaneous activities is important when system reliability is evaluated, fuzzy Petri networks are used as a final model. In fact, fuzzy Petri networks present graphical displaying from the system along with its mathematic solution.

2. PAPER ORGANIZATION

In the second section of the paper, some definitions have been presented like CaISR framework, fuzzy colored Petri networks, fuzzy enterprise architecture and enterprise reliability. In the third section, enterprise architecture reliability is evaluated. In the fourth section, a case study is carried out to clarity the paper idea. Finally, the paper is concluded.

2.1. CaIsR framework

CaIsR framework is based on the process of performing the work. This framework involves three perspectives involving operational, systemic and technical perspectives [5].

Operational perspective refers to describing tasks and activities, components and operational nodes, necessities of information transmission between the nodes, and they are necessary for performing or supporting operations. Systematic perspective is a description of systems and the communication between then, and they are considered for performing or supporting a task. Technical architecture perspective refers to determining minimum set of rules for sequence, performance and dependencies between the components or dements of a system whose purpose is to guarantee the requirements specified for that system [4].

2.2. Fuzzy colored Petri nets

A normal Petri net is compatible with the classic logic. In practice, they are complex systems, and there is some uncertainty. Hence, such systems are modeled with Petri nets, these uncertainties must be displayed with ambiguous and inaccurate statements in Petri models, and this requires introducing fuzzy concept in Petri model. Therefore, in 1988, various kinds of Petri nets are designed by the research carried out by Looney and his colleagues in terms of Petri nets and artificial intelligence association, and it is compatible with Petri net theory [6]. Different tools support this network such as artifex and Design/CPN, CPN Tools

software. The software of CPN tools is presented by Aarhus university of Denmark. The first version of this software was presented in the market in 2001. The language used for developing and correcting data is standard ML [2]. By using graphic user interface of this software, models of colored Petri nets, can be designed. Also, by using this animating software (step-by-step execution) in models of colored Petri nets as well as creating a space of colored Petri nets model, question definitions of model behavior and creating output files to displaying simulation results of models can be presented.

2.3. Fuzzy enterprise architecture

Fuzzy enterprise architecture (FEA) is a complete and comprehensive model describing and displaying enterprise architecture products by considering uncertainly in process and data of enterprise architecture. The model of fuzzy enterprise architecture (FEA) shows architecture products in the form of new frames and formats. In this way, the process of enterprise architecture designing can be obtained. Through using this model, enterprise architecture can be evaluated by Fuzzy colored Petri nets [7]. Also, the model of fuzzy enterprise architecture involves capability of enterprise architecture structure and describing behavioral concepts.

2.4. The reliability of enterprise architecture

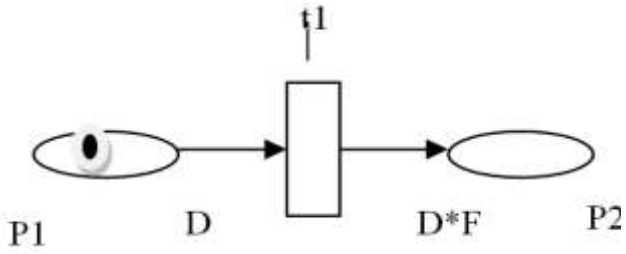
In enterprise architecture level, reliability is used to displaying reliability degrees of appropriate functions in all enterprise processes or appropriate functions and applications of a set of components in specified time period. It is important to mention that if a component fails (or it does not function appropriately), then they does not mean that while enterprise architecture is unreliable. Each component, in fact, indicates a part of an organization where it provides services. These components (services) meet final targets of the organization when they cooperate with each other.

In order to evaluate the reliability of an organization, reliability evaluation of each component is proposed. Evaluation of reliability in whole enterprise is the result of evaluating reliability of each component and the way of interacting and communicating with each other during execution of a process. In this research, the purpose is to evaluate reliability in a process of enterprise architecture.

3. EVALUATION OF ENTERPRISE ARCHITECTURE RELIABILITY

In this paper, accurate computation of whole enterprise architecture reliability is disregarded, and the way of computing reliability in a process is explained. In order to reach this purpose, (ov-6c) product that shows the procedures of interacting different components when a process is executed is firstly converted to fuzzy colored Petri net. Finally, in order to compute reliability of each process, a success degree of f is defined for each transmission in fuzzy Petri net. Success degree specifies the ability of activating transmission. If it occurs erroneously component output is not corrects, this event will not occur. In other words, error in 1-f,and it means missing data.

It is supposed that taken in transmission entry of it carries a value, and this shows the success degree to that place; that is process execution is successful in D degree. when transmission is activated, values showing success degree of (D) are changed to D*f, and token obtains a new activation and fire probability. D*f is used instead f f, and this concept is shown in figure (1) [8]. In the first step of the process (Petri net) , the success degree equals one.



D: the location of maintaining success rate= reliability

Figure (1) : Reliability by using fuzzy Petri net

It means that if, during process execution, reliability is equal to D in P1 component, it changes to D*f when passes t1. This procedure is repeated until while process is executed. With regard to this fact that the process passes through some components, reliability will be different.

4. CASE STUDY AND EVALUATION OF PROPOSED METHOD.

In order to classify discussion and proposed idea, a system is designed and tested. It's likely that generalization of this method requires using it in many real systems so that probable defects are removed. In this research, the amount of reliability and error is considered the same in all components. Hence, differences of processes reliability are the results of the number of procedures. In other words, a process passing through more components, its reliability will considerably decrease.

4.1. explaining assumed system of hotel reservation

A record system of centralized request is a computerized program used for recording rents and managing payments in a hotel. Here, a hotel reservation system based on web is selected. Organizations or other services transfer their own request to this service when hotel reservation is needed. According to various conclusions of system, it announces possibility of reservation. Such systems must have higher error tolerance [8]. The product of describing event/operational tracing (OV-6C) and event/systemic tracing (SU-loc) in hotel reservation hotel has been shown in figure (2). In this product, since the hotel response to applicant enterprise is fuzzy, fuzzy product (OV-6C) is considered.

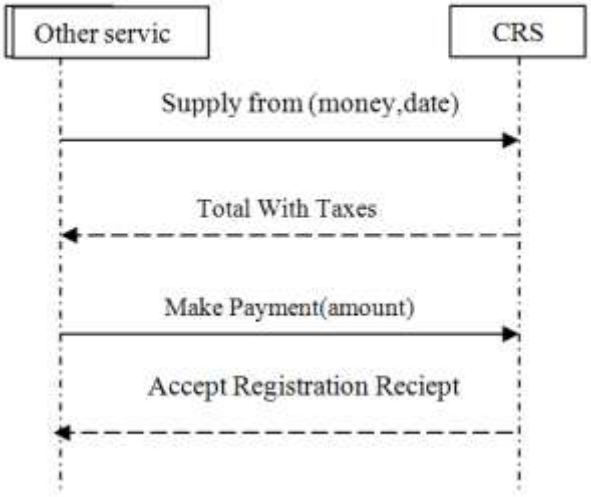


Figure (2): The product of describing event/operational

tracing (OV-6C) and event/systemic tracing (SV-10C) for hotel reservation system [8].

In order to present fuzzy Petri nets in hotel representation hotel, conditions, events and rules must be specified in the first step.

Step1:

Events

table (1): Rules

1. Money

- ✓ Almost enough, Money>500
- ✓ Middle 200<Money<500
- ✓ Not enough Money<200

2. Date

- ✓ Valid (reservation date for more than 10 days)
- ✓ Invalid (reservation date for less than 10 days)

3. Output

- ✓ Almost acceptable (acceptance is performed)
- ✓ Acceptable (acceptance is performed somewhat. If a person requests a room fpr five days, it is reserved for three days).
- ✓ Not acceptable (acceptance is not performed)

Rule	Event	Condition	State
R1	P12 :money is not enough P15: Date is invalid	P12 AND p15	P18: Not acceptable
R2	P12: Money is not enough P14: Date is valid	P12 AND P14	P18: Not acceptable
R3	P11: Money is enough P15: Date is invalid	P11 AND P15	P17: Acceptable
R4	P11: Money is enough P14: Date is valid	P11 AND P14	P16: Almost acceptable
R5	P13: Money is middle P15: Date is invalid	P13 AND P15	P17: Acceptable
R6	P13: Money is middle P14: Date is valid	P13 AND P14	P17: Acceptable

Finally, the final result of fuzzy Petri net in hotel reservation system is shown in figure (3).

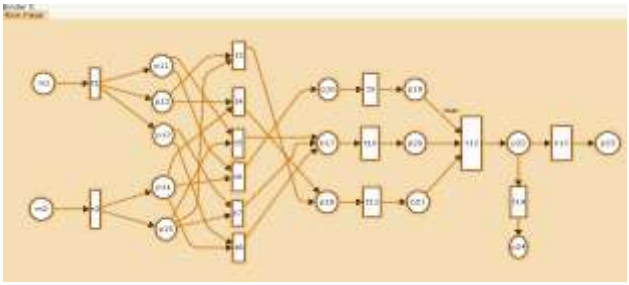


Figure3: Colored fuzzy Petri net of hotel reservation system

As it is observed in figure 3, if reliability is higher in P22, it is transferred to P23; Otherwise, it is transferred to P24, and it comes to end.

4.2. Numerical evaluation of supposed system

After preparing fuzzy Petri net, reliability is evaluated in the network.

Example 1: it is supposed that money=400 (average value of money) and 20th day of future month (that is, day is valid because it involves more than10 days. with regard to these inputs, the route is as follows:

t1,t2→t8→t10→t12→t13 or t x

It is supposed that all transmissions are fired with probability of 0.94 (that is, reliability of each component is 0.94), except t1 and t2 whose probable transmissions are equal to one.

The method of computing reliability

t1,t2=1

t3,t4,t5,t7,t8=0.94

t10= 0.94*0.94=0.88

t12=0.94*0.88=0.82

t13, t14= 0.94*0.82=0.77

Final reliability is 0.77, and this is very low because capability of each transmission is considered as 0.94. Therefore, two methods are proposed to create reliability of whole enterprise architecture:

1) Increasing reliability of each component (service)

2) By signing, the number of procedures or transmissions (components decreases in while enterprise architecture because reliability decreases in passing through each transmission (component) in enterprise architecture. In other words, the process having more procedures has lower reliability since in the process passing through each component, the reliability is multiplied by success rate of that component. As it is clear, success rate or reliability of a component is never 100% (one).

5. CONCLUSION

Various methods have been proposed to evaluate enterprise architecture, but none of them can describe and evaluate enterprise architecture when the enterprise has indecisive data or processes. In this research, enterprise architecture is evaluated for the first time by considering indecisive and fuzzy conditions. Also, fuzzy formal methods having strong mathematical support are used in this research for evaluation, and this indicated accuracy and reliability of the proposed method. This method is executable, and it has great value and importance because, by applying it on several real systems, its probable defects can be removed, and it can be used by enterprise architects.

The method proposed in this research is used to evaluate reliability of a component in enterprise architecture. Through developing this method in all components of enterprise architecture and combining and aggregating all of them, a complete and comprehensive method can be obtained for whole enterprise architecture, and this purpose can be achieved in future researches. Since this model uses formal models such as Petri net, it is accurate, and it can be

used by enterprise architects. Defects of this model can be completely removed by accurate investigation of this model and by using it in supposed enterprise architecture.

In addition, in order to evaluate other enterprise architecture metrics, a similar method (using fuzzy Petri net) can be used, and it will be discussed in future researches. The results of comparing the proposed model with other models of enterprise architecture evaluation

Evaluation criterion	Architecture model	Levis model	OSAN model	The proposed mode
application and function	No	Yes	Yes	Yes
supporting object	No	Yes	Yes	Yes
Evaluating qualitative feature	efficiency	efficiency	efficiency	efficiency
Obtaining the rates involving bottleneck components	No	No	No	No
Evaluating eterprise architecture by considering fuzzy condition	No	No	No	Yes

6. REFERENCES

[1] Research core of information systems architecture, 2011, Shahid Beheshti university, http://isa.sbu.ac.ir

[2] Javadpour, R, 2006. Presenting an executable model to evaluate enterprise architecture by using colored Petri nets. Misc thesis, shahid Beheshti university

[3] Iacob, ME., Jonkers, H., "Quantitative Analysis of Enterprise Architectures", Proc. INTEROP-ESA Conference, Geneva, pp.23–25, 2005.

[4] C4ISR Architecture Framework version 2.0. Office of the Assistant Secretary of Defense For Command, Control, Communications and Intelligence, Washington D.C., November 1997.

[5] DoD Architecture Framework., (2003), version 1.0, DoD architecture Framework Working Group, Department of Defense.

[6] Janette Cardoso and B.pradin-chezalviel, "logic and fuzzy petri net ".

[7] Afshani, j., Harounabadi, A., Abbasi Dezfouli, M.,"A new model for designing uncertain enterprise architecture", journal of Management Science Letters, Vol 2, No 2,2012, pp. 689-696.

[8] Nematzadeh, H., Safaai Bin Deris, Maleki, H., Nematzadeh,Z. ,"Evaluating Reliability of System Sequence Diagram Using Fuzzy Petri Net" International Journal of Recent Trends in Engineering, Issue. 1, Vol. 1, May 2009, pp. 142-147.

Subtitles

1-OperationalEvent/Trace Description
2- Fuzzy Logic Architecture

International Journal of Computer Applications Technology and Research
Volume 5– Issue 1, 08 - 14, 2016, ISSN:- 2319–8656

Feature Extraction Techniques and Classification Algorithms for EEG Signals to detect Human Stress - A Review

Chetan Umale
MIT College of Engineering,
Pune, India

Amit Vaidya
MIT College of Engineering,
Pune, India

Shubham Shirude
MIT College of Engineering,
Pune, India

Akshay Raut
MIT College of Engineering,
Pune, India

Abstract: EEG (Electroencephalogram) signal is a neuro signal which is generated due the different electrical activities in the brain. Different types of electrical activities correspond to different states of the brain. Every physical activity of a person is due to some activity in the brain which in turn generates an electrical signal. These signals can be captured and processed to get the useful information that can be used in early detection of some mental diseases. This paper focus on the usefulness of EGG signal in detecting the human stress levels. It also includes the comparison of various preprocessing algorithms (DCT and DWT.) and various classification algorithms (LDA, Naive Bayes and ANN.). The paper proposes a system which will process the EEG signal and by applying the combination of classifiers, will detect the human stress levels.

Keywords: Stress, DWT, KNN, LDA, Naïve Bayes, EEG signal, NeuroSky Mindwave.

1. INTRODUCTION

Stress is generally defined as a response of a person to the environmental demands or pressures. It results from interaction between a person and his/her environment that are perceived as straining or exceeding their adaptive capacities and threatening their well-being. It can be understood from above definition that stress is part and parcel of today's life style. If ignored, stress can lead to chronic diseases. The risk factors for stress related diseases are a mixture of personal, interpersonal and social variables. Thus, it can affect various phases of our life. So it is necessary to detect stress at an early stage and take appropriate measures.

Now, the question is, Is it possible to detect stress at early stages? Yes, it is. This is done by many psychologists or counselors. But it requires active participation from the person seeking counseling. This might not be possible in some cases when a stressed person is unable to express himself frankly. It makes the job of a counselor difficult . This problem can be solved, if the brain signals are recorded and analyzed to detect stress.

Brain signals are neuron signals. The electrical activity of the neurons inside the brain cause electric potential to be generated across different parts of the brain. The difference between these electric potential levels can be captured and used for various applications including stress detection. These brain signals are called as EEG signals - Electroencephalogram signal. Different types of states of the brain are due to different types of electrical activities of brain neurons. So, different signal values correspond to different mental states.

These signals can be captured using various available equipments which generally consists of electrodes which are placed on the scalp with a conductive gel between the electrodes and the scalp. Electrodes are placed at different positions on the scalp which capture the signals from different parts of the brain. Raw EEG signals cannot be used directly for stress detection. Pre-processing is required to extract useful features which can further used with various machine learning algorithms.

The aim of the paper is to review various feature extraction techniques and classification algorithms which can be used for detection of stress levels. Based on the review, a system is proposed which will use a single electrode EEG headset(Neurosky MindWave) to record raw EEG signals which will be pre-processed using Discrete Wavelet Transform(DWT) and classified using a combination of classifiers approach to detect stress levels. Section 1 gives an introduction on how EEG signals can be used in detection of stress. Section 2 contains literature survey. Section 3 contains the proposed system for stress detection. Section 4 is conclusion and Section 5 includes future scope.

2. LITERATURE SURVEY

2.1 EEG

Electroencephalography (EEG) is nothing but recorded electrical activity generated by brain [2].The first report on electrical brain activity in humans was published in 1929 which allowed doctors and scientists to observe the brain in action in a meaningful way [5]. There are millions of neurons in our brain. These activities generate millions of small electric voltage fields. The aggregate of these voltage fields can be detected by electrodes placed on the scalp. Thus we can say that, EEG is the superposition of many smaller signals. The

amplitude of these signals ranges from 1 µV to 100 µV in a normal person.

The different electrical frequencies in EEG can be associated with different physical actions and mental states [3]. So EEG shows a wide variations in amplitude depending on external stimulation and different internal mental states. The different frequency bands are Delta, theta, alpha, beta and gamma. Frequency bands associated with the different mental states are given in the Table 1.

Table *1*: EEG frequency bands

Brainwave type	Frequency range(Hz)	Mental states and conditions
Delta	0.1 to 3	Deep, dreamless sleep, unconscious
Theta	4 to 7	Intuitive, creative, recall, fantasy
Alpha	8 to 12	Relaxed but not drowsy, tranquil
Low Beta	12 to 15	Formerly SMR, relaxed yet focused
Midrange Beta	16 to 20	Thinking, aware of self & surrounding
High Beta	21 to 30	Alertness, agitation
Gamma	30 to 100	Motor functions, higher mental activity

2.2 SIGNAL ACQUIRING METHODS

Various methods are used for acquiring EEG signals. They differ in the way the electrodes are placed. The methods can be categorized as follows:

a) Invasive: Invasive EEG recordings are those recordings that are captured with electrodes that are placed on the surface or within the depth of the brain[6]. These type of methods are generally used in medical surgeries or implants. Again the two types of electrodes used in this method are

i) Subdural EEG electrodes: Subdural EEG electrodes are the electrodes which sit over the surface of the brain. The placement of these electrodes is often confirmed with co-registration on an MRI scan image.

ii) Depth EEG electrodes: Depth EEG electrodes are those which are placed within the substance of the brain.

b) Non-invasive: Non-invasive EEG recordings are those that are captured with electrodes that are placed on the scalp rather than placing it on the surface or within the depth of the brain. Electrodes used here are small metal discs which are made of stainless steel, tin, gold or silver covered with a silver chloride coating. They are placed on the scalp in special positions. These positions are specified using the International 10/20 system. Each electrode site is labeled with a letter and a number. The letter refers to the area of brain underlying the electrode e.g. F-Frontal lobe and T - Temporal lobe. Even numbers denote the right side of the head and odd numbers the left side of the head [7]. A mapping of these electrode positions according to the 10-20 international system is shown in the Figure 1.

This system is used for multichannel electrode system which are generally used for research purpose and are complex and not portable. Alternative to such systems is single electrode system which uses single channel for recording signal and are simpler and portable. An example of such system is NeuroSky MindWave headset. The comparison between the signals captured by single channel NeuroSky MindWave and multichannel Biopac system is. The red line is Biopac and a blue line is NeuroSky.

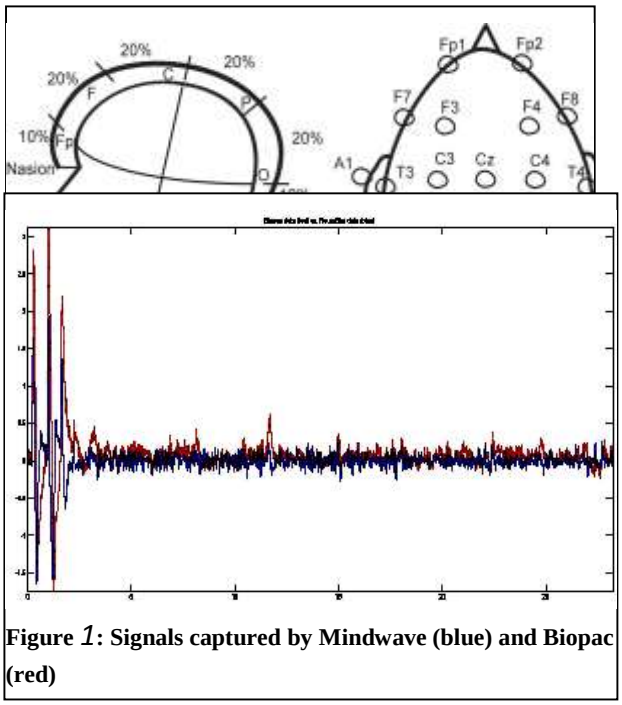


Figure *1*: Signals captured by Mindwave (blue) and Biopac (red)

2.3 MATLAB

MATLAB (Matrix Laboratory) is a multi-paradigm numerical computing environment and high level programming language developed by Math works. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python[9].EEG signals being electrical signals are vulnerable to outside interference and artifacts. Using signal processing capabilities of MATLAB, these problems can be resolved effectively. MATLAB provides an interactive toolbox called EEGLAB, which is effective for continuous and event-related EEG signals analysis using independent component analysis (ICA), time/frequency analysis (TFA), as well as standard averaging methods [10]. EEGLAB supports loading of existing EEG datasets as well as real time EEG datasets through software like Neuroscan.

2.4 Pre-processing

EEG signals are non-stationary and non-linear. EEG signals are susceptible to noise and interference caused by eye movement and muscle movement. The electronic devices in the vicinity can also cause interference. Also, the amount of raw data required for classification is impractical for most machine learning algorithms. Thus feature extraction is necessary for successful classification. Pre-processing includes transformation of EEG signals from time domain to frequency domain and removal of noise and artifacts.

A variety of feature extraction methods exist for BCI applications, such as Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT).

2.4.1 Discrete Cosine Transform (DCT)

Discrete Cosine Transform is a method to convert time series signals into frequency components. In context of BCI, DCT is used to calculate maximum, minimum and mean value of EEG signal. The one-dimensional DCT for a list of N real numbers is expressed by the following formula:

$$Y(u) = \sqrt{\frac{2}{N}} a(u) \sum_{x=0}^{N-1} f(x) \cos\left(\frac{\pi(2x+1).u}{2N}\right)$$

where

$$a(j) = \frac{1}{\sqrt{2}} \text{ if } j=0$$

$$a(j) = 1 \text{ if } j \neq 0$$

The input is a set of N data values (EEG samples) and the output is a set of N DCT transform coefficients Y(u). The first coefficient Y(0) is called the DC coefficient and it holds average signal value.. The rest coefficients are referred to as the AC coefficients [11]. DCT produces concentrated signals, where the energy is concentrated into few coefficients. Thus DCT is effective for data compression which leads to reduced size of input vector for machine learning algorithms and also reduces time required for learning.

2.4.2 Discrete Wavelet Transform (DWT)

Wavelet transform is the process of expressing any general function as an infinite series of wavelets. The main idea behind wavelet analysis is of expressing a signal as a linear combination of the particular set of functions by shifting and expanding the original wavelet (mother wavelet). This decomposition gives a set of coefficients called as wavelet coefficients, due to this the signal can be reconstructed as a linear combination of the wavelet functions weighted by the wavelet coefficients. The main feature of the wavelets is that most of their energy is restricted to a finite time interval .This is called as time-frequency localization. Frequency localization means that the Fourier transform is band limited. This time-frequency localization provides good frequency localization at low frequencies and good time localization at high frequencies. This produces segmentation of the time-frequency plane that is appropriate for most physical signals, especially those of a transient nature. This transform when applied to EEG signal will reveal features that are transient in nature [8].

Discrete wavelet transform depends on low pass filter g and high pass filter h. The working is based on two important functions namely wavelet function $\varphi_{i,l}(k)$ and scale function $\psi_{i,l}(k)$ which can be defined as **:**

$$\varphi_{i,l}(k)=2^{i/2}\, g_i(k-2^i l)$$

$$\psi_{i,l}(k)=2^{i/2}\, h_i(k-2^i l)$$

where the factor $2^{i/2}$ is an inner product normalization, i and l are the scale parameter and the translation parameter, respectively. The DWT decomposition is described as:

$$a_{(i)}(l)=x(k)*\varphi_{i,l}(k)$$

$$d_{(i)}(l)=x(k)*\psi_{i,l}(k)$$

where $a_{(i)}(l)$ is the approximation coefficient and $d_{(i)}(l)$ is the detail coefficient at resolution i.

The DWT decomposition of the input signal into different frequency bands is obtained by consecutive high-pass and low-pass filtering of the time domain signal. This decomposition is shown in the Figure 3.

In the given diagram, x[n] is the mother wavelet, h[n] is the high pass filter and g[n] is the low pass filter. EEG signals do not have any useful frequency components above 30 Hz [8].So the decomposition levels can be selected as 5. So the final relevant wavelet decomposition will be obtained at level A5.Sample EEG signal decomposition is shown in figure 4.

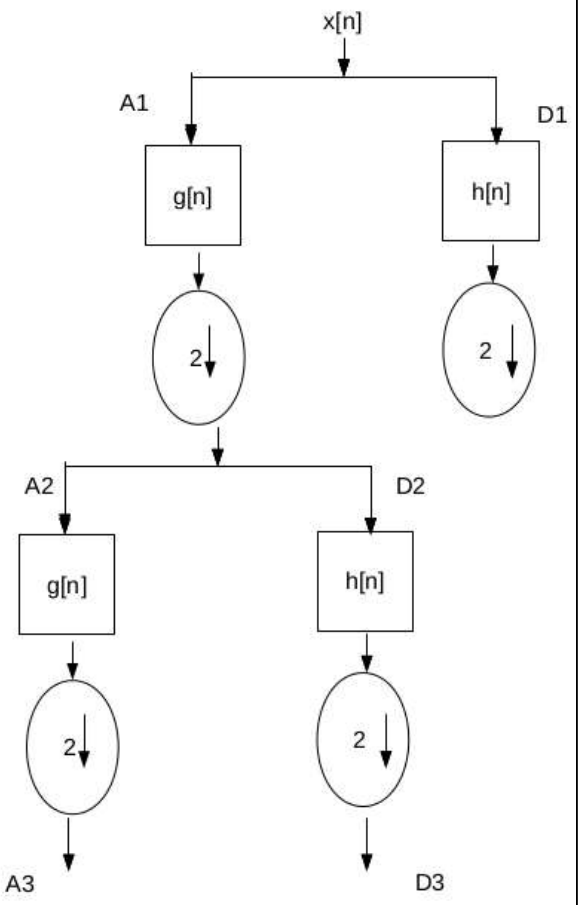


Figure 3: Decomposition of signal using DWT

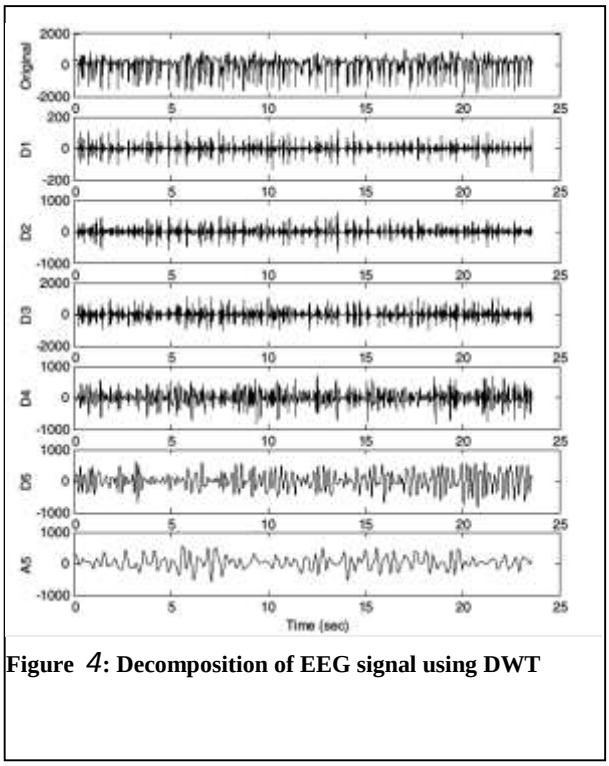


Figure 4: Decomposition of EEG signal using DWT

2.5 CLASSIFICATION

2.5.1 K-Nearest Neighbor (KNN)

K-nearest neighbor is an instance-based, lazy and supervised learning algorithm. KNN is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure. KNN is a non-parametric method that classifies the data by comparing the training data and testing data based on estimating the feature values[12]. These feature values are calculated by using the distance function such as Euclidean distance which is not difficult if the given parameter values are numeric. An object is then classified by the majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. The value of the k refers to how many nearest values should be considered before the output class is decided.

A commonly used distance metric is the Euclidean distance. The Euclidean distance of two points or tuples, say, $X_1 = (x_{11}, x_{12}, \ldots\ldots, x_{1n})$ and $X_2 = (x_{21}, x_{22}, \ldots\ldots, x_{2n})$, is

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$$

where, x_{1i} and x_{2i} represents the training and testing data respectively[13]. Different attributes are measured on different scales, so if the Euclidean distance formula is used directly, the effect of some attributes might be completely dwarfed by others that have larger scales of measurement.

After feature extraction process the EEG training data and test data is passed to the classification process. Then Euclidean distance is calculated between each EEG training sample and testing sample. The class for first K neighbors is considered and the majority vote is the classified class. The accuracy for the KNN is high as compared to the other classifiers.

2.5.2 Linear Discriminant Analysis (LDA)

Linear discriminant analysis (LDA) is one of the most popular classification algorithms for Brain Computer Interface applications, and has been used successfully in a large number of systems. LDA linearly transforms data from high dimensional space to low dimensional space. Finally, the decision is made in the low dimensional space. Thus the definition of the decision boundary plays an important role in classification process. During this process, the class distributions having some finite variance will be still kept in the projected space. Hence, we assume that if the mean and variance of the projected data is considered for the calculation of the decision boundary, it may extend LDA method to deal

with the practical heteroscedastic distribution data, which derives Z-LDA.

Consider the case of two classes (x11, x12, x13, …… , x1m) € C1 and (x21, x22, x23, …… , x2n) € C2 where m and n being number of training samples.

X=(x11, x12, x13, …… , x1m, x21, x22, x23,……, x2n) be our input sample.
Calculate weight sum y(X) by, y(X)

$$y(\vec{X}) = \vec{W}^T \vec{X}$$

where WT is weight vector.

Now, the parameters related to Gaussian distribution mean(μ) and standard deviation(σ) are calculated as :-

$$\mu_1 = \frac{1}{m}\sum_{x \in C_1} y(\mathbf{x})$$

$$\mu_2 = \frac{1}{n}\sum_{x \in C_2} y(\mathbf{x})$$

$$\sigma_1 = \sqrt{\frac{1}{m}\sum_{x \in C_1} (y(\mathbf{x}) - \mu_1)^2}$$

$$\sigma_2 = \sqrt{\frac{1}{n}\sum_{x \in C_2} (y(\mathbf{x}) - \mu_2)^2}$$

where μ1, μ2 and σ1, σ2 and mean and standard deviations of two training set samples C_K (K=1,2).

During classification process, when any sample X is input, first calculate weight sum y(x) and then perform following normalization procedure:-

z1=(y(x) - μ1)/ro1

z2=(y(x) - μ2)/ro2

where z1 and z2 are z-scores to calculate how much weight sums of given input sample is close to the training samples. Finally, the larger value among these z scores gives the final classification class for the input. That means, if z1>z2, the sample is going to be classified in class c1; otherwise c2.

2.5.3 Naive Bayes

Probability can be interpreted from two views: Objective and Subjective. The Subjective probability is called as Bayesian Probability. Bayesian Probability is the process for using the probability for predicting the likelihood of certain events occurring in the future. Naive Bayes is a conditional probability model where Bayes' theorem is used to infer the probability of hypothesis under the observed data or evidence [14]. Bayes theorem states that

$$\text{posterior} = \frac{\text{prior} * \text{likelihood}}{\text{evidence}}$$

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}$$

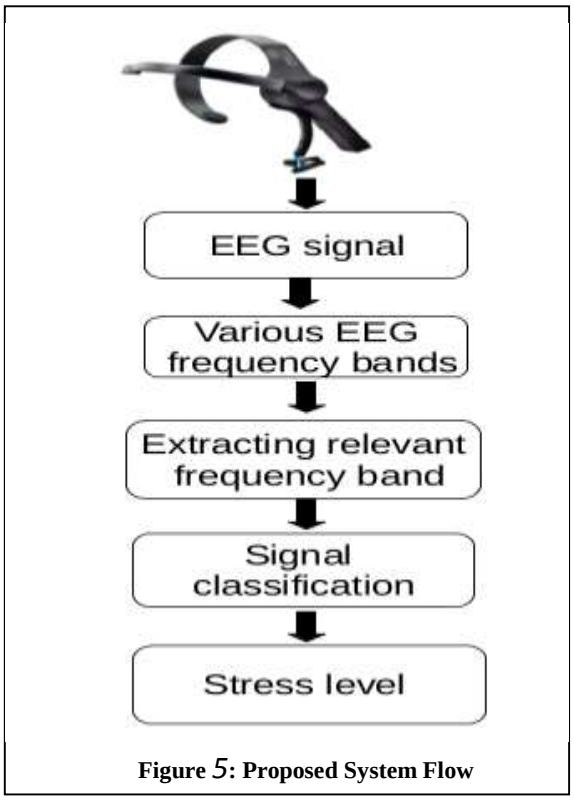


Figure 5: Proposed System Flow

When dealing with continuous data generated from EEG signals, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. First the data is segment by the class, and then compute the mean and variance of x in each class. Let μ_c be the mean of the values in x associated with class *c*, and let σ_c^2 be the variance of the values in x associated with class *c*. Then, the probability distribution of some value given a class, p(x = v|c), can be computed by plugging v into the equation for a Normal distribution parameterized by μ_c and σ_c^2. That equation is :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(v-\mu)^2}{2\sigma^2}}$$

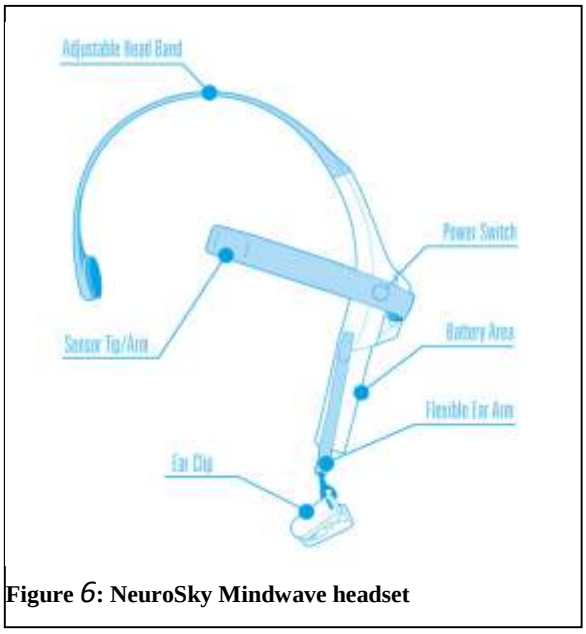


Figure 6: NeuroSky Mindwave headset

3. PROPOSED SYSTEM3.1 Data collection equipment

Here the data required is the EEG signal. In the proposed system, the equipment to be used in the data collection is NeuroSky Mindwave headset which uses single electrode for capturing the EEG signal. There are various noise sources in this process such as muscle movement or any electrical device in the vicinity. Primary filtering of these noise signals is done by the ear clip which acts as reference or ground.

3.2 System flow

Figure 6 shows the general flow of the proposed system. EEG signal is captured using the NeuroSky Mindwave. Processing is done in the MATLAB environment using the EEGLAB interface. EEGLAB provides functions to capture data in numerous scenarios such as at different sampling rate. The captured signal contains various EEG frequency bands. The frequency range relevant for stress level detection is 4 to 40 Hz [2]. Thus, only these frequencies are extracted. This process is called feature extraction. In the proposed system, DWT is used to preprocess the data as it has the unique advantage of time-frequency localization [8].

Once the preprocessing is done, next task is to classify the signal into appropriate stress level. When a single classifier is used, it is difficult to identify the misclassification error. The solution to this problem is to develop a more complex classifier with a little misclassification rate. This approach may seem promising but it will make the system complex. Another approach is to use combination of simple classifiers rather than using one complex classifier. One more advantage of this approach is that even if one of the classifier misclassifies the data, the other classifiers can rectify this error.

In the proposed system, an approach of combination of classifiers is used. The classifiers which are to be uses are KNN, Naive Bayes and LDA. The output class will be decided by voting and the class which gets majority of votes will be the output class.

4. CONCLUSION

Early detection of stress can help in prevention of chronic mental illness. Recording and analyzing EEG signals can be an effective tool for this purpose. Different feature extraction techniques and classification algorithms for EEG signal analysis are discussed in this paper. This review suggests use of single channel EEG headset which provides a portable and affordable alternative to traditional multichannel equipments.

5. FUTURE WORK

The paper proposed the methodology for detecting human stress level in real time. As mentioned, early detection and treatment of stress is important. Music therapy is a good option for stress treatment. Music with appropriate frequency can be played automatically corresponding to detected stress level. Another future application of EEG signal can be a biometric authentication system, as pattern of EEG signal captured from every person is unique.

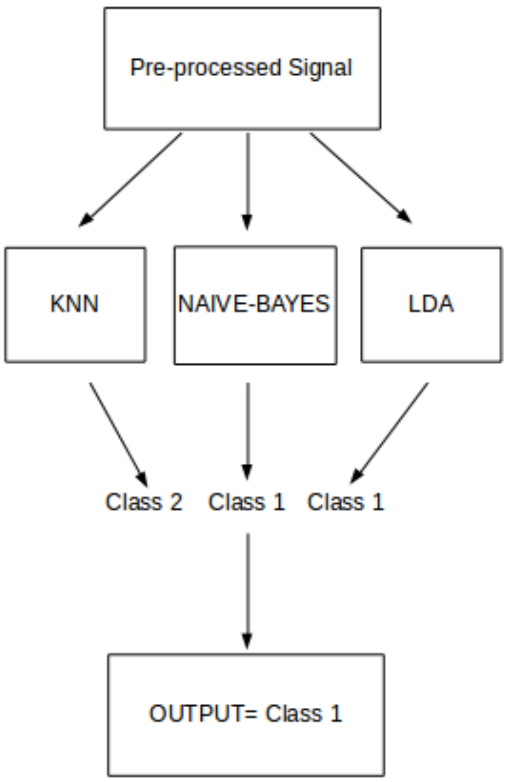


Figure 7: Combination of classifiers

6. REFERENCES

[1] http://www.stress.org.uk/what-is-stress.aspx

[2] NeuroSky Inc. Brain Wave Signal (EEG) of NeuroSky, Inc.(December 15, 2009). [Online]. Available:http://frontiernerds.com/files/neurosky-vs-medical-eeg.pdf

[3] Erik Andreas Larsen, "Classification of EEG Signals in a Brain-Computer Interface System" ,Norwegian University of Science and Technology Department of Computer and Information Science,June 2011

[4] D. Puthankattil Subha, Paul K. Joseph, Rajendra Acharya U, Choo Min Lim, "EEG Signal Analysis: A Survey" ,J Med Syst (2010) 34:195–212

[5] D. A. Kaiser. What is quantitative EEG. [Online].Available:http://www.skiltopo.com/skil3/what-is-qeeg-by-kaiser.pdf

[6] https://my.clevelandclinic.org/services/neurological_institute/epilepsy/diagnostics-testing/invasive-eeg-monitoring

[7] http://www.medicine.mcgill.ca/physio/vlab/biomed_signals/eeg_n.htm

[8] Abdulhamit Subasi, "EEG signal classification using wavelet feature extraction and a mixture of expert model",Expert Systems with Applications 32 (2007) 1084–1093,2006 Elsevier Ltd.

[9] https://en.wikipedia.org/wiki/MATLAB

[10] http://sccn.ucsd.edu/eeglab/

[11] Darius Birvinskas,Vacius Jusas,Ignas Martišius, Robertas Damaševičius, "Data Compression of EEG Signals for Artificial Neural Network Classification",ISSN 2335–884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2013, Vol.42, No.3

[12] Tatiur Rahman, Apu Kumer Ghosh, Md. Maruf Hossain Shuvo, Md. Mostafizur Rahman, "Mental Stress Recognition using K-Nearest Neighbor(KNN) Classifier on EEG Signals", International Conference on Materials, Electronics & Information Engineering, ICMEIE-2015.

[13] Chee-Keong Alfred Lim and Wai Chong Chia, "Analysis of Single-Electrode EEG Rhythms Using MATLAB to Elicit Correlation with Cognitive Stress", International Journal of Computer Theory and Engineering, Vol. 7, No. 2, April 2015

[14] Juliano Machado Alexandre Balbinot, Adalberto Schuck, "A study of the Naive Bayes classifier for analysing imaginary movement EEG signals using the Periodogram as spectral estimator", Bio-signals and Bio-robotics Conference (BRC), 2013 ISSNIP, IEEE conference publication.

International Journal of Computer Applications Technology and Research
Volume 5–Issue 1, 15-19, 2016, ISSN:2319–8656

Presenting a New Ant Colony Optimization Algorithm (ACO) for Efficient Job Scheduling in Grid Environment

Firoozeh Ghazipour
Department of Computer
Science and Research Branch
Islamic Azad University, Kish, Iran

Seyyed Javad Mirabedini
Department of Computer
Islamic Azad University
Central Tehran Branch

Ali Harounabadi
Department of Computer
Islamic Azad University
Central Tehran Branch

Abstract: Grid computing utilizes the distributed heterogeneous resources in order to support complicated computing problems. Job scheduling in computing grid is a very important problem. To utilize grids efficiently, we need a good job scheduling algorithm to assign jobs to resources in grids.
In the natural environment, the ants have a tremendous ability to team up to find an optimal path to food resources. An ant algorithm simulates the behavior of ants. In this paper, a new Ant Colony Optimization (ACO) algorithm is proposed for job scheduling in the Grid environment. The main contribution of this paper is to minimize the makespan of a given set of jobs. Compared with the other job scheduling algorithms, the proposed algorithm can outperform them according to the experimental results.

Keywords: jobs, scheduling, Grid environment, Ant Colony Optimization (ACO), makespan.

1. INTRODUCTION

Current scientific problems are very complex and need huge computing power and storage space. The past technologies such as distributed or parallel computing are unsuitable for current scientific problems with large amounts of data. Processing and storing massive volumes of data may take a very long time.
Grid computing [1] is a new paradigm for solving those complex problems. In grids, we need to consider the conditions such as network status and resources status. If the network or resources are unstable, jobs would be failed or the total computation time would be very large. So we need an efficient job scheduling algorithm for these problems in the grid environment.
The purpose of job scheduling is to balance the entire system load while completing all the jobs at hand as soon as possible according to the environment status. Because the environment status may change frequently, traditional job scheduling algorithm may not be suitable for the dynamic environment in grids.
In grids, users may face hundreds of thousands of computers to utilize. It is impossible for anyone to manually assign jobs to computing resources in grids. Therefore, grid job scheduling is a very important issue in grid computing. Because of its importance, importance, many job scheduling algorithms for grids [2-5] have been proposed.
A good schedule would adjust its scheduling strategy according to the changing status of the entire environment and the types of jobs. Therefore, a dynamic algorithm in job scheduling such as Ant Colony Optimization (ACO) [6, 7] is appropriate for grids.
ACO is a heuristic algorithm with efficient local search for combinatorial problems. ACO imitates the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths traveled. Many researches use ACO to solve NP-hard problems such as traveling salesman problem [8], graph coloring problem [9], vehicle routing problem [10], and so on.
This paper applies the ACO algorithm to job scheduling problems in Grid computing. We assume each job is an ant and the algorithm sends the ants to search for resources. We also modify the global and local pheromone update functions in ACO algorithm in order to balance the load for each grid resource. Finally, we compare the proposed ACO algorithm with Min-Min [11] and Max-Min [12]. According to the experimental results, we can find out that our proposed ACO algorithm is capable of achieving system load balance and decreasing the makespan better than other job scheduling algorithms. The rest of the paper is organized as follows. Section 2 explains the background of ACO algorithm and scheduling problem. Section 3 introduces some related work. Section 4 details the proposed ACO algorithm for job scheduling. Section 5 indicates the experimental results and finally, Section 6 concludes the paper.

2. BACKGROUND

2.1 Characteristics of ACO

Dorigo introduced the ant algorithm [18], which is a new heuristic algorithm and based on the behavior of real ants. When the blind insects, such as ants look for food, the moving ant lays some pheromone on the ground, thus marking the path it followed by a trail of this substance. While an isolated ant moves essentially at random, an ant encountering a previously laid trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The collective behavior that emerges means where the more are the ants following a trail, the more that trail becomes attractive for being followed. The process is thus characterized by a positive feedback loop, where the probability with which an ant chooses an optimum path increases with the number of ants that chose the same path in the preceding steps. Above observations inspired a new type of algorithm called ant algorithms or ant systems, which is presented by Dorigo and Gambardella [19]. The ACO algorithm uses a colony of artificial ants that behave as co-operative agents in a mathematical space were they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. Solution that satisfies the constraints is feasible. All ACO algorithms adopt specific algorithmic scheme which is shown in Figure 1.

```
Procedure ACO
    Parameter Initialization
    While (stopping criterion not satisfied) do
        Construct solutions
        Local search
        Update statistics
        Update pheromone trails
    End while
```

Figure 1. All ACO Algorithm scheme.

We utilize the characteristics of ant algorithms above mentioned to schedule job. We can carry on new job scheduling by experience depend on the result in the past job scheduling. It is very helpful for being within the grid environment.

2.2 Scheduling Problem Formulation

The grid environment consists of a large number of resources and jobs which should be assigned to the resources. The jobs cannot divide into smaller parts and after assigning a job to a resource, its executing cannot be stopped.

The main challenge in scheduling problems is time. Finding a solution in these problems tries to decrease the time of executing all jobs. In this case, the most popular criterion is makespan and our purpose in this paper is reducing the makespan with the aid of ACO.

In grid, we have a set of resources (Resources = $\{m_1, m_2, \ldots, m_m\}$) and a set of jobs (Jobs = $\{t_1, t_2, \ldots, t_n\}$) which should be assigned to the resources and executed on them. There is a matrix ETC [Resources] [Jobs] (Figure 2) that represents the time of executing (EX) t_i on m_j.

$$\begin{array}{c} \\ Resource_1 \\ Resource_2 \\ \vdots \\ Resource_m \end{array} \begin{array}{c} \begin{array}{cccc} Job_1 & Job_2 & \cdots & Job_n \end{array} \\ \begin{bmatrix} EX_{11} & EX_{12} & \cdots & EX_{1n} \\ EX_{12} & EX_{22} & \cdots & EX_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ EX_{m1} & EX_{m2} & \cdots & EX_{mn} \end{bmatrix} \end{array}$$

Figure 2. Matrix ETC

Suppose that $E_{ij\ (i \in Jobs\ ,\ j \in Resources)}$ is the time of job i on resource j and $W_{j\ (j \in Resources)}$ is the time of executing jobs which are assigned to resource j before. Then equation (1) shows the time of executing all jobs which are allocated to m_j.

$$\sum_{\forall\ Job\ is\ allocated\ to\ Resource\ j} (E_{ij} + W_j) \quad (1)$$

We can find the value of makespan according to equation (2):

$$\text{makespan} = \max \{\sum_{\forall\ job\ is\ allocated\ to\ resource\ j} (E_{ij} + W_j)\}, i \in Jobs\ and\ j \in Resources \quad (2)$$

With another look on these definitions, we can calculate the completion time of resources with equation (3):

$$CT_{ij} = E_{ij} + W_j \quad (3)$$

There is a Scheduling List for all resources that shows the jobs which are assigned to each resource. Each resource has a completion time and according to equation (4), the value of makespan is equal to the maximum completion time.

$$\text{makespan} = max_{\ (i,j) \in Scheduling\ List} (CT_{ij}) \quad (4)$$

The makespan is a criterion to evaluate the grid system and the main purpose of a scheduler is to minimize this criterion.

3. RELATED WORK

Jobs submitted to a grid computing system need to be processed by the available resources.

Best resources are categorized as optimal resources. In a research by [13], Ant Colony Optimization (ACO) has been used as an effective algorithm in solving the scheduling problem in grid computing.

ACO has been applied in solving many problems in scheduling such as Job Shop Problem, Open Shop Problem, Permutation Flow Shop Problem, Single Machine Total Tardiness Problem, Single Machine Total Weighted Tardiness Problem, Resource Constraints Project Scheduling Problem, Group Shop Problem and Single Machine Total Tardiness Problem with Sequence Dependent Setup Times [6]. A recent approach of ACO researches in the use of ACO for scheduling job in grid computing [14]. ACO algorithm has been used in grid computing because it is easily adapted to solve both static and dynamic combinatorial optimization problems and job scheduling in grid computing is an example.

Balanced job assignment based on ant algorithm for computing grids called BACO was proposed by [15]. The research aims to minimize the computation time of job executing in Taiwan UniGrid environment which focused on load balancing factors of each resource. By considering the resource status and the size of the given job, BACO algorithm chooses optimal resources to process the submitted jobs by applying the local and global pheromone update technique to balance the system load. Local pheromone update function updates the status of the selected resource after job has been assigned and the job scheduler depends on the newest information of the selected resource for the next job submission. Global pheromone update function updates the status of each resource for all jobs after the completion of the jobs. By using these two update techniques, the job scheduler will get the newest information of all resources for the next job submission. From the experimental result, BACO is capable of balancing the entire system load regardless of the size of the jobs. However, BACO was only tested in Taiwan UniGrid environment.

An ant colony optimization for dynamic job scheduling in grid environment was proposed by [16] which aimed to minimize the total job tardiness time. The initial pheromone value of each resource is based on expected execution time and actual execution time of each job. The process to update the pheromone value on each resource is based on local update and global update rules as in ACS. In that study, ACO algorithm performed the best when compared to First Come First Serve, Minimal Tardiness Earliest Due Date and Minimal Tardiness Earliest Release Date techniques.

The study to improved ant algorithm for job scheduling in grid computing which is based on the basic idea of ACO was proposed by [17]. The pheromone update function in this research is performed by adding encouragement, punishment coefficient and load balancing factor. The initial pheromone value of each resource is based on its status where job is assigned to the resource with the maximum pheromone value. The strength of pheromone of each resource will be updated after

completion of the job. The encouragement and punishment and local balancing factor coefficient are defined by users and are used to update pheromone values of resources. If a resource completed a job successfully, more pheromone will be added by the encouragement coefficient in order to be selected for the next job execution. If a resource failed to complete a job, it will be punished by adding less pheromone value. The load of each resource is taken into account and the balancing factor is also applied to change the pheromone value of each resource.

4. THE PROPOSED ALGORITHM

Real ants foraging for food lay down quantities of pheromone (chemical substance) marking the path that they follow. An isolated ant moves essentially at random but an ant encountering a previously laid pheromone will detect it and decide to follow it with high probability and thereby reinforce it with a further quantity of pheromone. The repetition of the above mechanism represents the auto catalytic behavior of real ant colony where the more the ants follow a trail, the more attractive that trail becomes [13].

The ACO algorithm uses a colony of artificial ants that behave as co-operative agents in a mathematical space were they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. Solution that satisfies the constraints is feasible. After initialization of the pheromone trails, ants construct feasible solutions, starting from random nodes, and then the pheromone trails are updated. At each step ants compute a set of feasible moves and select the best one (according to some probabilistic rules) to carry out the rest of the tour. The transition probability is based on the heuristic information and pheromone trail level of the move. The higher value of the pheromone and the heuristic information, the more profitable it is to select this move and resume the search.

4.1 Initialize Pheromone and Probability List

At the beginning, a population of ants is generated and they start their own search from one of the resource (the ants assigned to resources randomly). The initial Pheromone and Probability List is set to a small positive value t_0 and then ants update this value after completing the construction stage. In the nature there is not any pheromone on the ground at the beginning, or the initial pheromone in the nature is $t_0 = 0$. If in ACO algorithm the initial Pheromone is zero, then the probability to choose the next state will be zero and the search process will stop from the beginning. Thus it is important to set the initial Pheromone and Probability List to a positive value. The value of t0 is calculated with equation (5):

$$t_0 = \frac{1}{Jobs \times Resources} \quad (5)$$

4.2 Local Pheromone Update

Moving from a state to another means that a job is assigned to a resource. After choosing next node by ants, the pheromone trail should be updated. This update is local pheromone update and the equation (6) shows how it happens:

$$\text{Pheromone}_{ij}^{k} = \left((1-\varepsilon) \times \left(\text{Pheromone}_{ij}^{k}\right)\right) + (\varepsilon \times \theta) \quad (6)$$

In local pheromone update equation, θ is a coefficient which obtains its value from equation (7):

$$\theta = \frac{t_0}{CT_{ij}(Ant_k)} \quad (7)$$

The less value of CT_{ij}, the more value of θ. In fact, if the value of θ is larger, the more pheromone value will be deposited. Therefore, the chance of choosing resource j in next assigning is more than other resources.

4.3 Probability List Update

In addition to update Pheromone, the Probability List should be updated, too. The ants choose the next states based on heuristic information, equation (8):

$$\text{Heuristic}_{ij}^{k} = \frac{1}{(W_j) \times (ETC_{ij})} \quad (8)$$

With the heuristic information, we can update the Probability List, equation (9):

$$\text{Probability List}_{ij}^{k} = (\text{Pheromone}_{ij}^{k}) \times (\text{Heuristic}_{ij}^{k})^{\beta} \quad (9)$$

4.4 Global Pheromone Update

In the nature, some pheromone value on the trails evaporates. At the end of each iteration in the proposed algorithm, when all ants finish the search process, the all ants' value of pheromone will be reduced by evaporation rule, equation (10):

$$\text{Pheromone}_{ij}^{k} = (\text{Pheromone}_{ij}^{k}) \times (1-\rho) \quad (10)$$

When all ants construct a solution, it means that the ants moves from the nest to the food resource and finish the search process (all the jobs are assigned to the resources in grid). In the proposed algorithm, the best solution and the best ant which construct that solution will be found. The global pheromone update is just for the ant that finds the best solution. This ant is the best ant of iteration. At this stage, the value of Pheromone should be updated, equation (11):

$$\text{Pheromone}_{ij}^{Best\ Ant} = \text{Pheromone}_{ij}^{Best\ Ant} + \left((\rho) \times (\Delta)\right) + \frac{\varepsilon}{makespan\ (Best\ Ant)} \quad (11)$$

In global pheromone update, ρ is the elitism coefficient and Δ is calculated by equation (12):

$$\Delta = \frac{1}{makespan\ (Best\ Ant)} \quad (12)$$

The pseudo-code of the proposed algorithm is presented in Figure 3.

5. EXPERIMENTAL RESULTS

The results of the evaluation of the proposed algorithm with the two algorithms of Min-Min and Max-Min [11] for scheduling independent jobs in grid environment are presented in this section.

All experiments have been done on a system running Windows 7 Professional operating system with configuration of 2 GHz CPU and 2GB of RAM.

The Proposed ACO Algorithm for Job Scheduling in Grid
Parameters Initialization (Table 1)
Create Ant Population
Initialize the Pheromone and Probability Information (Formula 5)
While (stopping criterion not satisfied) **do**
  Construct Solution Phase
    For all Ants **do**
      Place all Ants in random chosen Resource
      Apply Local Update for (Ant, Job, Resource) (Formula 6, 9)
    End For
    While (all Jobs not assigned) **do**
      Apply Decision Rules for (Ant, Job)
      Apply Local Update for (Ant, Job, Resource) (Formula 6, 9)
    End While
  End of Construct Solution Phase
  Global Pheromone Update Phase
    Evaporate (Formula 10)
    Find and Update best Ant of Iteration and best-so-far
    Deposit Pheromone (Formula 11)
  End of Global Pheromone Update Phase
  Find best solution "best makespan"
  Reset the knowledge of all Ants
End While

Figure 3. Pseudo-code of the proposed algorithm.

Table 1 indicates the amounts of parameters which are used in executing the proposed algorithm.

Table 1. Parameters of the proposed algorithm.

Number of resources	16
Number of jobs	512
Number of ants	100
Number of iterations	1000
ε	0.1
β	2 (between 2 and 5)
ρ	0.2

A real heterogeneous computational system such as grid is a combination of hardware and software elements and a comparison of the scheduling techniques is often complicated in this environment. To solve this problem, Braun et al. [20], proposed a simulation model. They defined a grid environment which consists of a set of resources and a set of independent jobs. The scheduling algorithms aim to minimize the makespan. All scheduling algorithms need to know the completion time of each job on each resource. The model consists of 12 different kinds of examples: u_c_hihi, u_c_hilo, u_c_lohi, u_c_lolo, u_i_hihi, u_i_hilo, u_i_lohi, u_i_lolo, u_s_hihi, u_s_hilo, u_s_lohi, u_s_lolo; that any of them can be shown in a matrix. This model uses a matrix ETC which illustrates the estimated times of completion (Figure 2).
In this paper, the same model is used to evaluate the proposed algorithm and the two scheduling algorithms, Min-Min and Max-Min. After executing these three algorithms, different amounts of makespan are obtained which are shown in Table 2.

Table 2. Comparison of three algorithms' makespans.

	The Proposed Algorithm	Min-Min	Max-Min
u_c_hihi	8291837.50	8460674.00	12385672.00
u_c_hilo	158629.16	161805.42	204054.58
u_c_lohi	265301.53	275837.34	392566.69
u_c_lolo	5388.70	5441.43	6945.36
u_i_hihi	3085218.25	3513919.25	8018377.50
u_i_hilo	77190.69	80755.68	151923.84
u_i_lohi	108951.52	120517.71	251528.84
u_i_lolo	2617.01	2785.65	5177.71
u_s_hihi	4776037.50	5160343.00	9208811.00
u_s_hilo	102499.90	104375.17	172822.69
u_s_lohi	134873.02	140284.50	282085.69
u_s_lolo	3590.33	3806.83	6232.24

The Results of experiment are shown as a chart in Figure 4, 5, 6 and 7.

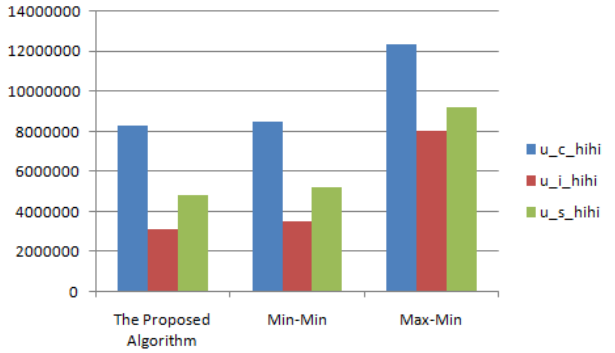


Figure 4. Algorithms' makespans based on u-*-hihi.

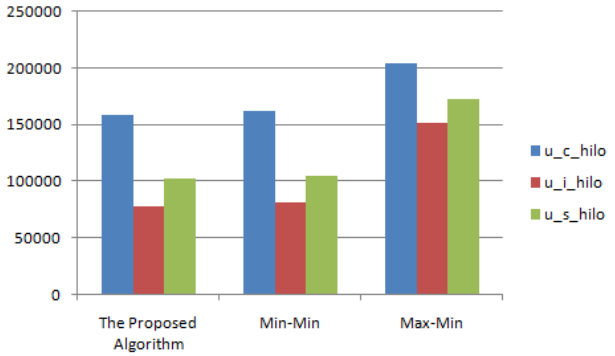


Figure 5. Algorithms' makespans based on u-*-hilo.

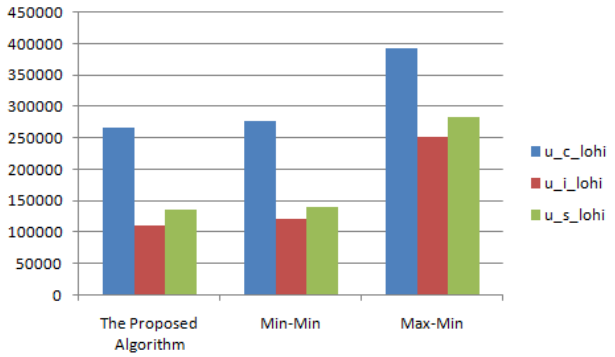


Figure 6. Algorithms' makespans based on u-*-lohi.

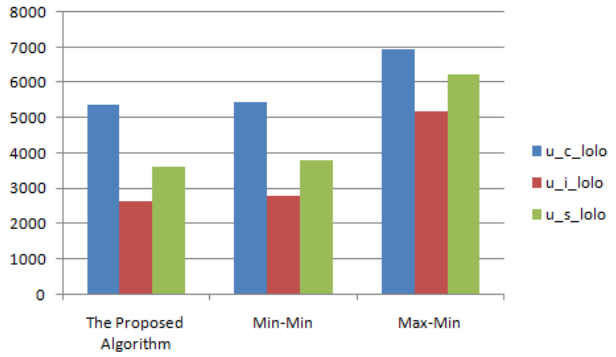


Figure 7. Algorithms' makespans based on u-*-lolo.

The two scheduling algorithms, Min-Min and Max-Min, are two best algorithms among other scheduling algorithms but the results of the experiment (Table 2) indicate that the proposed algorithm has higher performance and lower amount of makespan than the other two scheduling algorithms.

6. CONCLUSIONS

In this paper, a new ACO algorithm is proposed to choose suitable resources to execute jobs according to the completion times of resources and the size of given job in the grid environment. The local and global pheromone update functions are changed to do balance the system load. Local pheromone update function updates the status of the selected resource after jobs assignment. Global pheromone update function updates the status of scheduling list of best solution. The purpose of this paper is to minimize the makespan and the experimental results show that the proposed algorithm is capable of minimizing the makespan better than other two scheduling algorithms.

7. REFERENCES

[1] Reed, D. A., "Grids, the TeraGrid and beyond", IEEE, Vol. 36, Issue 1, 2003.

[2] Chang, R. S., Chang, J. S., Lin, P. S., "An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems, Vol. 25, pp. 20-27, 2009.

[3] Ku-Mahamud, K. R., Nasir, H. J. A., "Ant Colony Algorithm for Job Scheduling in Grid Computing", Fourth Asia International Conference on Mathematical/Analytical Modeling and Computer Simulation, pp. 40-45, 2010.

[4] Chang, R. S., Chang, J. S., Lin, S. Y., "Job scheduling and data replication on data grids", Future Generation Computer Systems, Vol. 23, Issue 7, pp. 846-860, 2007.

[5] Gao, Y., Rong, H., Huang, J. Z., "Adaptive grid job scheduling with genetic algorithms", Future Generation Computer Systems, Vol. 21, Issue 1, pp. 151-161, 2005.

[6] Dorigo, M., Stutzle, T., "Ant Colony Optimization", A Bradford Book, 2004.

[7] Dorigo, M., Blum, C., "Ant colony optimization theory: A survey", Theoretical Computer Science, Vol. 344, Issue 2, pp. 243-278, 2005.

[8] Dorigo, M., Gambardella, L. M., "Ant colony system: a cooperative learning approach to the traveling salesman problem", IEEE Transaction on Evolutionary Computation, Vol. 1, Issue 1, pp. 53-66, 1997.

[9] Salari, E., Eshghi, K., "An ACO algorithm for graph coloring problem", ICSC Congress on Computational Intelligence Methods and Applications, 2005.

[10] Zhang, X., Tang, L., "CT-ACO-hybridizing ant colony optimization with cyclic transfer search for the vehicle routing problem", ICSC Congress on Computational Intelligence Methods and Applications, 2005.

[11] Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., Freund, R. F., "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems", Journal of Parallel and Distributed Computing, pp. 107-131, 1999.

[12] Casanova, H., Legrand, A., Zagorodnov, D., Berman, F., "Heuristics for scheduling parameter sweep applications in grid environments", Heterogeneous Computing Workshop, pp. 349-363, 2000.

[13] Fidanova, S., Durchova, M., "Ant Algorithm for Grid Scheduling Problem", Springer, pp. 405-412, 2006.

[14] Pavani, G. S., Waldman, H., "Grid Resource Management by means of Ant Colony Optimization", 3rd International Conference on Broadband Communications, Networks and Systems, pp. 1-9, 2006.

[15] Chang, R. S., Chang, J. S., Lin, P. S., "Balanced Job Assignment Based on Ant Algorithm for Computing Grids", The 2nd IEEE Asia-Pacific Service Computing Conference, pp. 291-295, 2007.

[16] Lorpunmanee, S., Sap, M. N., Abdullah, A. H., Chompoo-inwai, C., "An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment", Proceedings of World Academy of Science, Engineering and Technology, Vol. 23, pp. 314-321, 2007.

[17] Yan, H., Shen, X. Q., Li, X., Wu, M. H., "An improved ant algorithm for job scheduling in grid computing", Proceedings of the fourth International Conference on Machine Learning and Cybernetics, Vol. 5, pp. 2957-2961, 2005.

[18] Dorigo, M., Maniezzo, V., Colorni, A., "Ant system: optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man and Cybernetics - Part B, Vol. 26, No. 1, pp. 1-13, 1996.

[19] Dorigo, M., Gambardella, L. M., "Ant colonies for the traveling salesman problem", Biosystems, Vol. 43, Issue 2, pp. 73-81, 1997.

[20] Braun, T. D., Siegel, H. J., Beck, N., "A Comparison of Eleven static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing systems", Journal of Parallel and Distributed Computing, Vol. 61, pp. 810-837, 2001.

International Journal of Computer Applications Technology and Research
Volume 5–Issue 1, 20-25, 2016, ISSN:2319–8656

Proposing a new Job Scheduling Algorithm in Grid Environment Using a Combination of Ant Colony Optimization Algorithm (ACO) and Suffrage

Firoozeh Ghazipour
Department of Computer
Science and Research Branch
Islamic Azad University, Kish, Iran

Seyyed Javad Mirabedini
Department of Computer
Islamic Azad University
Central Tehran Branch, Iran

Ali Harounabadi
Department of Computer
Islamic Azad University
Central Tehran Branch,Iran

Abstract: Scheduling jobs to resources in grid computing is complicated due to the distributed and heterogeneous nature of the resources. The purpose of job scheduling in grid environment is to achieve high system throughput and minimize the execution time of applications. The complexity of scheduling problem increases with the size of the grid and becomes highly difficult to solve effectively.
To obtain a good and efficient method to solve scheduling problems in grid, a new area of research is implemented. In this paper, a job scheduling algorithm is proposed to assign jobs to available resources in grid environment. The proposed algorithm is based on Ant Colony Optimization (ACO) algorithm. This algorithm is combined with one of the best scheduling algorithm, Suffrage. This paper uses the result of Suffrage in proposed ACO algorithm. The main contribution of this work is to minimize the makespan of a given set of jobs. The experimental results show that the proposed algorithm can lead to significant performance in grid environment.

Keywords: jobs, scheduling, Grid environment, Ant Colony Optimization (ACO), Suffrage, makespan.

1. INTRODUCTION

Distributed systems consist of multiple computers that communicate through computer networks. Research by [1] defined that cluster and grid computing are the most suitable ways for establishing distributed systems. Cluster computing environment consists of several personal computers or workstations that combined through local networks in order to develop distributed applications. However, applications are difficult to be flexible in cluster computing because they are limited to a fixed area. Grid computing is proposed to overcome this problem where various resources from different geographic area are combined in order to develop a grid computing environment. The study by [2] defined that grid computing is based on large scale resources sharing in a widely connected network such as the Internet.
The past technologies such as cluster and parallel computing do not suit current scientific problems with a large amount of data files [3]. Especially, processing and storing massive volumes of data may take a very long time. Besides, we need to consider about the other conditions such as network status and resources status. If the network or resources are unstable, jobs would be failed or the total computation time would be very large. So we need an efficient job scheduling algorithm for these problems in the grid environment.
How to schedule jobs efficiently in a grid environment is a main issue. The purpose of job scheduling is to balance the entire system load and minimize the completion time according to the environment status. A good scheduler would adjust its scheduling strategy according to the changing status of the entire environment and types of jobs.
In grid computing environment, there exists more than one resource to process jobs. One of the main challenges is to find the best or optimal resources to process a particular job in term of minimizing the job computational time. Computational time is a measure of how long that resource takes to complete the job. An effective job scheduling algorithm is needed to reduce the computational time of each resource and also minimize the makespan of the system. Therefore, an algorithm in job scheduling such as Ant Colony Optimization (ACO) [4] is appropriate for the problems mentioned above.
ACO is a heuristic algorithm with efficient local search for combinatorial problems. ACO imitates the behavior of real ant colonies in nature to search food from nest and interact with each other by pheromone value which is laid on paths. Many researches use ACO to solve NP-hard problems such as traveling salesman problem [5], graph coloring problem [6], vehicle routing problem [7], and so on.
One of the best scheduling algorithms which used to schedule jobs in grid environment is Suffrage [8]. This paper combines the ACO and Suffrage to schedule a given set of jobs in Grid computing. We assume each job is an ant and the algorithm sends the ants to search for resources. We also modify the global and local pheromone update functions in ACO algorithm in order to balance the load for each grid resource. Finally, we compare the proposed combinatorial algorithm with Min-Min [9], Max-Min [9] and the Suffrage itself [8]. According to the experimental results, we can find out that our proposed ACO algorithm is capable of achieving system load balance and decreasing the makespan better than other job scheduling algorithms. The rest of the paper is organized as follows. Section 2 explains the background of ACO algorithm and scheduling problem and Suffrage. Section 3 introduces some related work. Section 4 details the proposed ACO algorithm for job scheduling in grid environment. Section 5 indicates the experimental results and finally, Section 6 concludes this paper.

2. BACKGROUND

2.1 Features of ACO

Dorigo introduced the ant algorithm [10], which is a new heuristic algorithm and based on the behavior of real ants. When the blind insects, such as ants look for food, the moving ant lays some pheromone on the ground, thus marking the path it followed by a trail of this substance. While an isolated ant moves essentially at random, an ant encountering a previously laid trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The collective behavior that emerges means where the more are the ants following a trail, the more that trail becomes attractive for being followed. The process is thus characterized by a positive feedback loop, where the probability with which an ant chooses an optimum path increases with the number of ants that chose the same path in the preceding steps. Above observations inspired a new type of algorithm called ant algorithms or ant systems, which is presented by Dorigo and Gambardella [11]. The ACO algorithm uses a colony of artificial ants that behave as co-operative agents in a mathematical space were they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. Solution that satisfies the constraints is feasible. All ACO algorithms adopt specific algorithmic scheme which is shown in Figure 1.

```
Procedure ACO
    Parameter Initialization
    While (stopping criterion not satisfied) do
        Construct solutions
        Local search
        Update statistics
        Update pheromone trails
    End while
```

Figure 1. All ACO Algorithm scheme.

We utilize the characteristics of ant algorithms above mentioned to schedule job. We can carry on new job scheduling by experience depend on the result in the past job scheduling. It is very helpful for being within the grid environment.

2.2 Scheduling Problem Formulation

The grid environment consists of a large number of resources and jobs which should be assigned to the resources. The jobs cannot divide into smaller parts and after assigning a job to a resource, its executing cannot be stopped.

The main challenge in scheduling problems is time. Finding a solution in these problems tries to decrease the time of executing all jobs. In this case, the most popular criterion is makespan and our purpose in this paper is reducing the makespan with the aid of ACO.

In grid, we have a set of resources (Resources = {m_1, m_2, …, m_m}) and a set of jobs (Jobs = {t_1, t_2, …, t_n}) which should be assigned to the resources and executed on them. There is a matrix ETC [Resources] [Jobs] (Figure 2) that represents the time of executing (EX) t_i on m_j.

$$\begin{array}{c} \\ Resource_1 \\ Resource_2 \\ \vdots \\ \vdots \\ Resource_m \end{array} \begin{array}{c} \begin{array}{cccc} Job_1 & Job_2 & \dots & Job_n \end{array} \\ \begin{bmatrix} EX_{11} & EX_{12} & \dots & EX_{1n} \\ EX_{12} & EX_{22} & \dots & EX_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ EX_{m1} & EX_{m2} & \dots & EX_{mn} \end{bmatrix} \end{array}$$

Figure 2. Matrix ETC

Suppose that $E_{ij\ (i \in Jobs\ ,\ j \in Resources)}$ is the time of job i on resource j and $W_{j\ (j \in Resources)}$ is the time of executing jobs which are assigned to resource j before. Then equation (1) shows the time of executing all jobs which are allocated to m_j.

$$\sum_{\forall\ Job\ is\ allocated\ to\ Resource\ j} (E_{ij} + W_j) \quad (1)$$

We can find the value of makespan according to equation (2):

$$\text{makespan} = \max \{\sum_{\forall\ job\ is\ allocated\ to\ resource\ j} (E_{ij} + W_j)\}, i \in Jobs\ and\ j \in Resources \quad (2)$$

With another look on these definitions, we can calculate the completion time of resources with equation (3):

$$CT_{ij} = E_{ij} + W_j \quad (3)$$

There is a Scheduling List for all resources that shows the jobs which are assigned to each resource. Each resource has a completion time and according to equation (4), the value of makespan is equal to the maximum completion time.

$$\text{makespan} = max_{\ (i,j) \in Scheduling\ List} (CT_{ij}) \quad (4)$$

The makespan is a criterion to evaluate the grid system and the main purpose of a scheduler is to minimize this criterion.

2.3 Suffrage

Suffrage is one of the job scheduling algorithms that schedules the given set of jobs in grid environment. The idea behind *Suffrage* [9] is that a resource is assigned to a job that would "suffer" the most if that resource would not be assigned to it. The suffrage value of a job ($Difference_{ij}$) is defined by the difference between its second best completion time (Second Minimum) and its best completion time (First Minimum), equation 5:

$$Difference_{ij}\ (t_i) = [\text{First Minimum}_{ij} - \text{Second Minimum}_{ij}] \quad (5)$$

The suffrage value ($Difference_{ij}$) of all jobs in the given set of jobs is calculated and stored in a Difference List. Then, a job is possibly assigned to the resource that has the most suffrage value (has the most Priority), equation 6:

$$\text{Priority} = max_{(i,j) \in Difference\ list} (Difference_{ij}\ (t_j)) \quad (6)$$

If another job was previously assigned to the resource, the suffrage values of the job previously assigned and of the new job are compared. The job would be executed that has the greater suffrage value and the other one would be assigned

later. Every time a job finishes, all jobs that have not started yet are unscheduled and the algorithm is invoked again, using current values of suffrage. Consequently, the algorithm runs again and schedules the remaining jobs, but this time with the new load of the resources. This scheme is repeated until all jobs are assigned to the available resources and completed [12].

3. RELATED WORK

Jobs submitted to a grid computing system need to be processed by the available resources.

Best resources are categorized as optimal resources. In a research by [13], Ant Colony Optimization (ACO) has been used as an effective algorithm in solving the scheduling problem in grid computing.

The study by [14] proposed a bio-inspired adaptive job scheduling mechanism in grid computing. The purpose of this research is to minimize the execution time of the computational jobs by effectively taking advantage of the large amount of distributed resource. Various software ant agents were designed with simple functionalities. The pheromone value of each resource depends on their execution time. Resource with high execution time will receive a large number of pheromone. In this research, the comparison was also performed between the bio inspired adaptive scheduling with the random mechanism and heuristic mechanism. Experimental results showed that a bio-inspired adaptive job scheduling has good adaptability and robustness in a dynamic computational grid.

Stutzle, T. proposes Max-Min Ant System (MMAS) [15] to limit the pheromone range to be greater than or equal to the low bound value (Min) and smaller than or equal to the upper bound value (Max) to avoid ants to converge too soon in some ranges.

M. Dorigo et al propose Fast Ant System (FANT) [16]. It uses one ant at each iteration and gets the solution of the ant to do a local search. FANT works without evaporation rule and it updates pheromone after each iteration. In order to avoid the sub-optimal solution, it applies the reset pheromone function.

Hui Yan et al [17] apply the basic idea of ACO, but change the pheromone update function by adding encouragement, punishment coefficient and local balancing factor. The initial pheromone value of each resource is based on its status. For example, the number of CPU, CPU speed and bandwidth will take into account on the initial pheromone value. They assign job to the resource with the maximum pheromone value and the pheromone of each resource will be update by update function. The encouragement and punishment and local balancing factor coefficient are defined by users and are used to update pheromone values of resources. If a resource completed a job successfully, it will be added more pheromone by the encouragement coefficient in order to be selected for next job execution. If a resource failed to complete a job, it will be punished by adding less pheromone value. They take the load of each resource into account and also apply the balancing factor to change the pheromone value of each resource.

Kwang Mong Sim et al. [18] use multiple kinds of ant to find multiple optimal paths for network routing. The idea can be applied to find multiple available resources to balance resources utilization in job scheduling. The key of the idea is each different kinds of ant can only sense their own kind of pheromone so that it can find much different paths including the shortest-path by different kinds of ant. Its main problem is that if all kinds of ant find the same path, it will do nothing like using one kind of ant. And how to compare their performance of each kind of ant creates another problem. Furthermore, one solution from this algorithm may work efficiently in an environment, but it may work inefficiently in another one.

J. Heinonen et al. [19] apply the hybrid ACO algorithm with different visibility to job-shop scheduling problem. The hybrid ACO algorithm consists of two ideas. One idea is the basic ACO algorithm, and the other idea uses the post-processing algorithm in the part of local search in ACO algorithm. When the ACO algorithm finished, all ants complete its own tour which can be decomposed into blocks. The block for swap must contain more than two operations. Then the post processing algorithm uses the swap operation on the blocks. If the swap reforms the makespan, the new path is accepted; otherwise the swap is invalid and the swapped block recovers to previous status.

Balanced job assignment based on ant algorithm for computing grids called BACO was proposed by [3]. The research aims to minimize the computation time of job executing in Taiwan UniGrid environment which focused on load balancing factors of each resource. By considering the resource status and the size of the given job, BACO algorithm chooses optimal resources to process the submitted jobs by applying the local and global pheromone update technique to balance the system load. Local pheromone update function updates the status of the selected resource after job has been assigned and the job scheduler depends on the newest information of the selected resource for the next job submission. Global pheromone update function updates the status of each resource for all jobs after the completion of the jobs. By using these two update techniques, the job scheduler will get the newest information of all resources for the next job submission. From the experimental result, BACO is capable of balancing the entire system load regardless of the size of the jobs. However, BACO was only tested in Taiwan UniGrid environment.

An ant colony optimization for dynamic job scheduling in grid environment was proposed by [20] which aimed to minimize the total job tardiness time. The initial pheromone value of each resource is based on expected execution time and actual execution time of each job. The process to update the pheromone value on each resource is based on local update and global update rules as in ACS. In that study, ACO algorithm performed the best when compared to First Come First Serve, Minimal Tardiness Earliest Due Date and Minimal Tardiness Earliest Release Date techniques.

4. THE PROPOSED ALGORITHM

Real ants foraging for food lay down quantities of pheromone (chemical substance) marking the path that they follow. An isolated ant moves randomly but an ant encountering a previously laid pheromone will detect it and decide to follow it with high probability and thereby reinforce the path with a further quantity of pheromone. The repetition of the above mechanism represents the auto catalytic behavior of real ant colony where the more the ants follow a trail, the more attractive that trail becomes [13].

The ACO algorithm uses a colony of artificial ants that behave as co-operative agents in a mathematical space were they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. Solution that satisfies the constraints is feasible. After initialization of the pheromone trails, ants construct feasible solutions, starting from random nodes, and then the pheromone trails are updated. At each step ants compute a set of feasible moves and select the best one (according to some probabilistic rules) to carry out the rest of the tour. The transition probability is based on the heuristic information and pheromone trail level of the move. The higher value of the pheromone and the heuristic information, the more profitable it is to select this move and resume the search.

4.1 Initialize Pheromone and Probability List

At the beginning, a population of ants is generated and they start their own search from one of the resource (the ants assigned to resources randomly). The initial Pheromone and Probability List is set to a small positive value t_0 and then ants update this value after completing the construction stage. In the nature there is not any pheromone on the ground at the beginning, or the initial pheromone in the nature is $t_0 = 0$. If in ACO algorithm the initial Pheromone is zero, then the probability to choose the next state will be zero and the search process will stop from the beginning. Thus it is important to set the initial Pheromone and Probability List to a positive value. The value of t0 is calculated with equation (7):

$$t_0 = \frac{1}{Jobs \times Resources} \quad (7)$$

4.2 Local Pheromone Update

Moving from a state to another means that a job is assigned to a resource. After choosing next node by ants, the pheromone trail should be updated. This update is local pheromone update and the equation (8) shows how it happens:

$$\text{Pheromone}_{ij}^{k} = \left((1-\varepsilon) \times \left(\text{Pheromone}_{ij}^{k}\right)\right) + (\varepsilon \times \theta) \quad (8)$$

In local pheromone update equation, θ is a coefficient which obtains its value from equation (9):

$$\theta = \frac{t_0}{CT_{ij}(Ant_k)} \quad (9)$$

The less value of CT_{ij}, the more value of θ. In fact, if the value of θ is larger, the more pheromone value will be deposited. Therefore, the chance of choosing resource j in next assigning is more than other resources.

4.3 Probability List Update

In addition to update Pheromone, the Probability List should be updated, too. The ants choose the next states based on heuristic information, equation (10):

$$\text{Heuristic}_{ij}^{k} = \frac{1}{(W_j) \times (ETC_{ij})} \quad (10)$$

With the heuristic information, we can update the Probability List, equation (11):

$$\text{Probability List}_{ij}^{k} = (\text{Pheromone}_{ij}^{k}) \times (\text{Heuristic}_{ij}^{k})^{\beta} \quad (11)$$

4.4 Global Pheromone Update

In the nature, some pheromone value on the trails evaporates. At the end of each iteration in the proposed algorithm, when all ants finish the search process, the all ants' value of pheromone will be reduced by evaporation rule, equation (12):

$$\text{Pheromone}_{ij}^{k} = (\text{Pheromone}_{ij}^{k}) \times (1-\rho) \quad (12)$$

When all ants construct a solution, it means that the ants moves from the nest to the food resource and finish the search process (all the jobs are assigned to the resources in grid). In the proposed algorithm, the best solution and the best ant which construct that solution will be found. The global pheromone update is just for the ant that finds the best solution. This ant is the best ant of iteration. At this stage, the value of Pheromone should be updated, equation (13):

$$\text{Pheromone}_{ij}^{Best\ Ant} = \text{Pheromone}_{ij}^{Best\ Ant} + \left((\rho) \times (\Delta)\right) + \frac{\varepsilon}{makespan\ (Best\ Ant)} \quad (13)$$

In global pheromone update, ρ is the elitism coefficient and Δ is calculated by equation (14):

$$\Delta = \frac{1}{makespan\ (Best\ Ant)} \quad (14)$$

The pseudo-code of the proposed algorithm is presented in Figure 3.

The Proposed ACO Algorithm for Job Scheduling in Grid
Parameters Initialization (Table 1)
Create Ant Population
Choose one best ant randomly among ant population
Initialize the Pheromone and Probability Information (Formula 7)
Set the one best ant to Suffrage
While (stopping criterion not satisfied) **do**
  Construct Solution Phase
    For all Ants **do**
      Place all Ants in random chosen Resource EXCEPT the one best
      Apply Local Update for (Ant, Job, Resource) (Formula 8, 11)
    End For
    While (all Jobs not assigned) **do**
      Apply Decision Rules for (Ant, Job)
      Apply Local Update for (Ant, Job, Resource) (Formula 8, 11)
    End While
  End of Construct Solution Phase
  Global Pheromone Update Phase
    Evaporate (Formula 12)
    Find and Update best Ant of Iteration and best-so-far
    Deposit Pheromone (Formula 13)
  End of Global Pheromone Update Phase
  Find best solution "best makespan"
  Reset the knowledge of all Ants
End While

Figure 3. Pseudo-code of the proposed algorithm.

5. EXPERIMENTAL RESULTS

The results of the evaluation of the proposed algorithm with the three algorithms of Min-Min and Max-Min [9] and Suffrage [8] for scheduling independent jobs in grid environment are presented in this section.

All experiments have been done on a system running Windows 7 Professional operating system with configuration of 2 GHz CPU and 2GB of RAM.

Table 1 indicates the amounts of parameters which are used in executing the proposed algorithm.

Table 1. Parameters of the proposed algorithm.

Number of resources	16
Number of jobs	512
Number of ants	100
Number of iterations	1000
ε	0.1
β	3 (between 2 and 5)
ρ	0.2

A real heterogeneous computational system such as grid is a combination of hardware and software elements and a comparison of the scheduling techniques is often complicated in this environment. To solve this problem, Braun et al. [21], proposed a simulation model. They defined a grid environment

which consists of a set of resources and a set of independent jobs. The scheduling algorithms aim to minimize the makespan. All scheduling algorithms need to know the completion time of each job on each resource. The model consists of 12 different kinds of examples: u_c_hihi, u_c_hilo, u_c_lohi, u_c_lolo, u_i_hihi, u_i_hilo, u_i_lohi, u_i_lolo, u_s_hihi, u_s_hilo, u_s_lohi, u_s_lolo; that any of them can be shown in a matrix. This model uses a matrix ETC which illustrates the estimated times of completion (Figure 2).

In this paper, the same model is used to evaluate the proposed algorithm and the three scheduling algorithms, Suffrage, Min-Min and Max-Min. After executing these four algorithms, different amounts of makespan are obtained which are shown in Table 2.

Table 2. Comparison of four algorithms' makespans.

	The Proposed Algorithm	Suffrage	Min-Min	Max-Min
u_c_hihi	8106103.00	10249174.00	8460674.00	12385672.00
u_c_hilo	158464.09	168982.61	161805.42	204054.58
u_c_lohi	264181.47	337121.44	275837.34	392566.69
u_c_lolo	5328.91	5658.54	5441.43	6945.36
u_i_hihi	3073985.00	3306819.25	3513919.25	8018377.50
u_i_hilo	75600.41	77589.09	80755.68	151923.84
u_i_lohi	108290.52	114578.91	120517.71	251528.84
u_i_lolo	2617.01	2639.32	2785.65	5177.71
u_s_hihi	4776037.50	5121953.50	5160343.00	9208811.00
u_s_hilo	102499.90	102499.88	104375.17	172822.69
u_s_lohi	134873.02	150297.11	140284.50	282085.69
u_s_lolo	3587.83	3846.47	3806.83	6232.24

The Results of experiment are shown as charts in Figure 4, 5, 6 and 7.

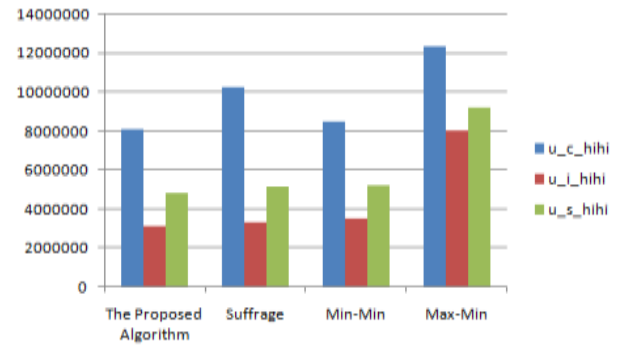


Figure 4. Algorithms' makespans based on u-*-hihi.

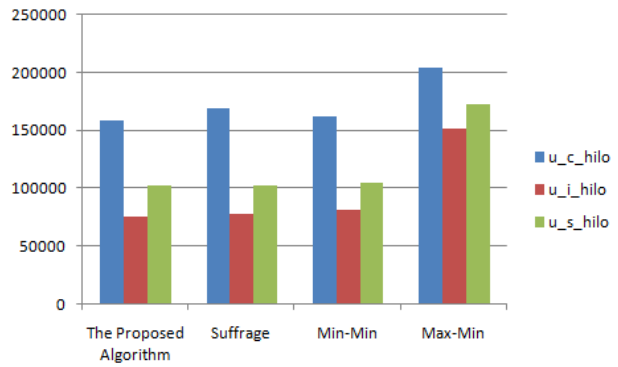


Figure 5. Algorithms' makespans based on u-*-hilo.

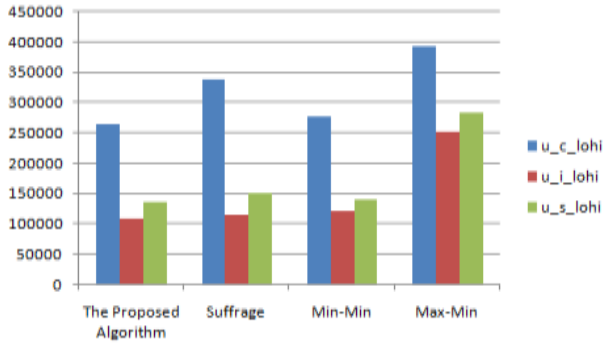


Figure 6. Algorithms' makespans based on u-*-lohi.

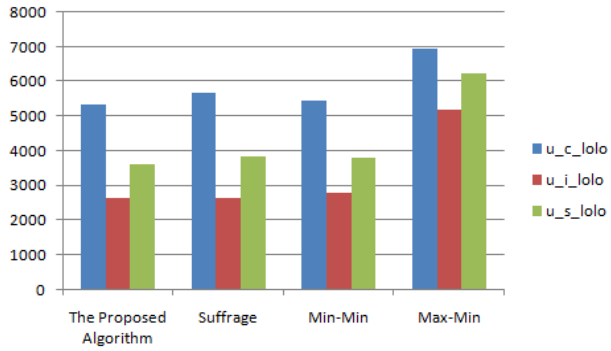


Figure 7. Algorithms' makespans based on u-*-lolo.

The three scheduling algorithms, Suffrage, Min-Min and Max-Min, are three best algorithms among other scheduling algorithms but the results of the experiment (Table 2) indicate that the proposed algorithm has higher performance and lower amount of makespan than the other three scheduling algorithms.

6. CONCLUSIONS

In this paper, a new ACO algorithm is proposed to choose suitable resources to execute jobs according to the completion times of resources and the size of given job in the grid environment. The proposed algorithm is a combination of ACO and Suffrage; which means that the result of Suffrage is used in proposed ACO. One ant as an elite one is set to Suffrage' solution. The local and global pheromone update functions are changed to do balance the system load. Local pheromone update function updates the status of the selected resource after jobs assignment. Global pheromone update function updates the status of scheduling list of best solution. The purpose of this paper is to minimize the makespan and the experimental results states that the proposed combinatorial algorithm is capable of minimizing the makespan better than other three scheduling algorithms.

7. REFERENCES

[1] Yan, K. Q., Wang, S. S., Wang, S. C., Chang, C. P., "Towards a hybrid load balancing policy in grid computing system", Expert Systems with Applications, Vol. 36, pp. 12054-12064, 2009.

[2] Yang, K., Guo, X., Galis, A., Yang, B., Liu, D., "Towards efficient resource on-demand in Grid Computing", ACM SIGOPS Operating Systems Review, Vol. 37, Issue 2, pp. 37-43, 2003.

[3] Chang, R. S., Chang, J. S., Lin, P. S., "Balanced Job Assignment Based on Ant Algorithm for Computing Grids", The 2nd IEEE Asia-Pacific Service Computing Conference, pp. 291-295, 2007.

[4] Dorigo, M., Blum, C., "Ant colony optimization theory: A survey", Theoretical Computer Science, Vol. 344, Issue 2, pp. 243-278, 2005.

[5] Dorigo, M., Gambardella, L. M., "Ant colony system: a cooperative learning approach to the traveling salesman problem", IEEE Transaction on Evolutionary Computation, Vol. 1, Issue 1, pp. 53-66, 1997.

[6] Salari, E., Eshghi, K., "An ACO algorithm for graph coloring problem", ICSC Congress on Computational Intelligence Methods and Applications, 2005.

[7] Zhang, X., Tang, L., "CT-ACO-hybridizing ant colony optimization with cyclic transfer search for the vehicle routing problem", ICSC Congress on Computational Intelligence Methods and Applications, 2005.

[8] Casanova, H., Legrand, A., Zagorodnov, D., Berman, F., "Heuristics for scheduling parameter sweep applications in grid environments", Heterogeneous Computing Workshop, pp. 349-363, 2000.

[9] Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., Freund, R. F., "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems", Journal of Parallel and Distributed Computing, pp. 107-131, 1999.

[10] Dorigo, M., Maniezzo, V., Colorni, A., "Ant system: optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man and Cybernetics - Part B, Vol. 26, No. 1, pp. 1-13, 1996.

[11] Dorigo, M., Gambardella, L. M., "Ant colonies for the traveling salesman problem", Biosystems, Vol. 43, Issue 2, pp. 73-81, 1997.

[12] Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., Freund, R. F., "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems", Journal of Parallel and Distributed Computing, pp. 107-131, 1999.

[13] Fidanova, S., Durchova, M., "Ant Algorithm for Grid Scheduling Problem", Springer, pp. 405-412, 2006.

[14] Li, Y., "A bio-inspired adaptive job scheduling mechanism on a computational grid", International Journal of Science and Network Security (IJCSNS), Vol. 6, Issue 3, pp. 1-7, 2006.

[15] Stutzle, T., "Max-Min Ant System for Quadratic Assignment Problems", Citeseer, 1997.

[16] Taillard, E. D., Gambardella, L. M., "Adaptive Memories for the Quadratic Assignment Problem", Citeseer, pp. 1-18, 1997.

[17] Yan, H., Shen, X. Q., Li, X., Wu, M. H., "An improved ant algorithm for job scheduling in grid computing", Proceedings of the fourth International Conference on Machine Learning and Cybernetics, Vol. 5, pp. 2957-2961, 2005.

[18] Sim, K. M., Sun, W. H., "Multiple ant-colony optimization for network routing", Proceedings of the First International Symposium on Cyber Worlds, pp. 277-281, 2002.

[19] Heinonen, J., Pettersson, F., "Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem", Applied Mathematics and Computation, Vol. 187, Issue 2, pp. 989-998, 2007.

[20] Lorpunmanee, S., Sap, M. N., Abdullah, A. H., Chompoo-inwai, C., "An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment", Proceedings of World Academy of Science, Engineering and Technology, Vol. 23, pp. 314-321, 2007.

[21] Braun, T. D., Siegel, H. J., Beck, N., "A Comparison of Eleven static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing systems", Journal of Parallel and Distributed Computing, Vol. 61, pp. 810-837, 2001.

International Journal of Computer Applications Technology and Research
Volume 5– Issue 1, 26 - 33, 2016, ISSN:- 2319–8656

GRID SEARCHING

Novel way of Searching 2D Array

Rehan Guha
Institute of Engineering & Management
Kolkata, India

Abstract: Linear/Sequential searching is the basic search algorithm used in data structures. Linear search is used to find a particular element in a 2D array. It is not compulsory to arrange an array in any order (Ascending or Descending) as in case of 2D binary search. In this paper, I present a unique searching algorithm named Grid Search, which helps to search an unsorted 2D Array/Matrix with least time complexity and iteration. We also have compared the Grid searching algorithm with Linear Search Algorithm. We used C++ for implementation and analysis of CPU time taken by both the algorithms. Results have shown that Grid Searching Algorithm is working well for all input values and it takes lesser time than Sequential Searching in all aspects.

Keywords: Matrix Searching algorithms; 2D Array Searching algorithms; 2D Array Linear Searching; 2D Array Grid Searching; Complexity Analysis; Algorithms; Searching Algorithms;

1. INTRODUCTION

2D Array or Matrix searching algorithm is a widely used and known searching algorithm. Though it is a primitive and basic searching technique, but it is widely used in different fields even today. Fields include Bio-medical image processing, Rasterization techniques, Electro-optical displays, Pixel tracing and many more.

In this paper I have developed a new searching technique named Grid Searching, based on the existing limitations of the 2D array linear/sequential search algorithm.

In 2D array searching an iterator[1] traverses sequentially from left to right (row major) or top to down (column major). When it encounters an element which is equal to the key[2] the search stops and the index of the element is returned, but if the key is not present in the 2D array the function will return 0, implying that the key is not found. Sequential search or linear search is the basic search algorithm used in data structures for 2D array[1] with unsorted data.

2. RELATED WORK

Linear search is used to find a particular element in an array. It is not compulsory to arrange an array in any order[3] as in the case of binary search. Linear search starts by sequentially scanning the elements in the 2D array (Row major wise or Column major wise) and if the element has been found, it will display the particular element's index value in the 2D array/matrix, else it will display not found[2][3].

If we declare a 2D array of order (n X n) i.e. size of array is n^2 then we initialize the array with the value (either static or dynamic declaration of values)[4]. In that if we mention the particular element to be identified, the linear search will start to search the array from the index 0, 0. If the value is not found, then the iterator will increment the index value by 1 in the column/row index becoming (0, 1) / (1, 0) and then it will check the element in that index again. This process continues till the element has been identified in the 2D Array/Matrix.

[1] It is a variable which acts as an counter variable and acts as an index for 2D array/matrix
[2] The element which is to be searched in the 2D array/matrix
[3] Ascending, Descending or any other sequence/series

Algorithm(s):

- LINEAR_SEARCH (M, NUM, n)

Input:

- 2D array/matrix M
- NUM is the key which is to be searched in M
- n is the order of the 2D array/matrix M (n X n)
- i & j are the iterators and works as index of the 2D array/matrix.

Pseudo Code: where, n= 1,2,3,4,5,6,7 …

LINEAR_SEARCH (M, NUM, n)

```
1.  Flag ← 0
2.  for i ←0 to n-1
3.          for j←0 to n-1
4.                  if (M[i][j] == NUM) then
5.                          Flag←1
6.                          return (i, j)
7.                  end if
8.          next j
9.  next i
10. if (Flag == 0) then
11.         return 0
12. end if
```

Flag –Checks whether the element is present or not.

If function LINEAR_SEARCH finds a match with the key then it returns the index of the element(s) (there might be multiple match of elements in the 2D array/matrix). Else it returns 0 which signifies element not found.

3. MODIFIED ALGORITHM (GRID SEARCHING)

If we declare and initialize a 2D array of order (n X n) i.e. size of the array is n^2 and if we mention the particular element to be identified, then the grid search will start searching the array from the index 1, 1. Now considering the index, it will search its boundary like a Grid. If the element is not found in the boundary then it will search the next set of Grids according to the order of matrix. This process continues till the element has been identified.

Algorithm(s):

- GRID_SEARCH(M, n, NUM),
- CHECK (M, n, i, j, NUM),
- RANGE (n)

Input:

- 2D array/matrix M
- NUM is the key which is to be searched in M
- n is the order of the 2D array/matrix M (n X n)
- i & j are the iterators and works as index of the 2D array/matrix.

Pseudo Code: where, n= 1,2,3,4,5,6,7 …

RANGE (n)

```
1.  if (n MOD 3 == 1) then
2.          T ←(n+2) / 3
3.  else if (n MOD 3 == 2) then
4.          T ←(n+1) / 3
5.  else
6.          T ←(n) / 3
7.  end if
8.  return ( T * T )
```

CHECK (M, n, i, j, NUM)

```
1.  if ( ((i>=0) AND (i<n)) AND ((j>=0) AND (j<n)) AND M[i][j] == NUM ) then
2.          return 1
3.  end if
4.  return 0
```

GRID_SEARCH (M, n, NUM)

```
1.  Flag ← 0
2.  T ←RANGE(n)
3.  for i←1 to T
4.          for j←1 to T
5.                  if (CHECK(M, n, i-1, j-1, NUM)) then
6.                          Flag ← 1
7.                          return(i-1, j-1)
8.          end if
```

```
9.                  if (CHECK(M, n, i-1, j, NUM)) then
10.                         Flag ← 1
11.                         return(i-1, j)
12.                 end if
13.                 if (CHECK(M, n, i, j-1, NUM)) then
14.                         Flag ← 1
15.                         return(i, j-1)
16.                 end if
17.                 if (CHECK(M, n, i-1, j+1, NUM)) then
18.                         Flag ← 1
19.                         return 1
20.                 end if
21.                 if (CHECK(M, n, i, j, NUM)) then
22.                         Flag ← 1
23.                         return(i, j)
24.                 end if
25.                 if (CHECK(M, n, i, j+1, NUM)) then
26.                         Flag ← 1
27.                         return (i, j+1)
28.                 End if
29.                 if (CHECK(M, n, i+1, j-1, NUM)) then
30.                         Flag ← 1
31.                         return(i+1, j-1)
32.                 End if
33.                 if (CHECK(M, n, i+1, j, NUM)) then
34.                         Flag ← 1
35.                         return(i+1, j)
36.                 end if
37.                 if (CHECK(M, n, i+1, j+1, NUM)) then
38.                         Flag ← 1
39.                         return(i+1, j+1)
40.                 end if
41.         next (j+3)
42. next (i+3)
43. if (Flag == 0) then
44.         return 0
45. end if
```

NOTE: Here the order of nested If-Else condition is important and cannot be changed. If it is changed the result of the algorithm may vary.

Example: If array M has 1 element only i.e. order of the array is 1(i.e. 1 X 1) and now if we search the array and the element is found then the total number of searches will equal to 1. But if the order of the nested If-Else condition is changed then the total number searches will be greater than 1 thus in turn increases the complexity of the algorithm.

Flag –Checks whether the element is present or not.

T –Holds the returned value from the function RANGE(n).

If function GRID_SEARCH finds a match with the key then it returns the index of the element(s) (there might be multiple match of elements in the 2D Array/Matrix). Else it returns 0 which signifies element is not found.

4. PROOF OF CORRECTNESS FOR GRID SEARCHING

To prove that an algorithm is correct, we need to show two things: (1) that the algorithm terminates, and (2) that it produces the correct output [5] [6].

4.1 Algorithm Grid Searching terminates after a finite number of steps

The variable T[4] holds a finite number according to the function RANGE(n) (refer Section 3) and the iterator variable starts from 1 and ends at T with increment of 3. If the iteration variable is greater than equal to T then the loop terminates and does not go into an infinite loop.

4.2 Algorithm Grid Searching produces correct output

If the element is found then the function GRID_SEARCH returns the index value of the found element(s) as well as sets the value of Flag[3] to 1 and if the value is NOT FOUND then the function GRID_SEARCH returns 0 indicating that the element is not present in the 2D array/matrix, thus resulting to a correct output.

[4] refer to Grid Searching algorithm under Section 3

5. PERFORMANCE ANALYSIS AND COMPARISON

Both the searching algorithm (Sequential Searching and Grid Searching) were implemented in C++ using GNU G++ Compiler. Both the searching algorithm was executed on machine with:-

32-bit Operating System having Intel® Core™ i3-4005U CPU @ 1.70GHz, 1.70GHz and installed memory (RAM) 4.00GB.

5.1 Analysis of the Function of both the algorithms

Function of Linear/Sequential searching *f(x)*:

$f(x) = x^2 + x + 1$ …..(i)

Function of Grid searching *g(x)*:

$g(x) = x^2/9 + 18x + 5$ …..(ii)

The functions are derived from their respective algorithms (refer Section 2 & 3). The function has taken the **for** loop(s), **if-else** condition(s) into consideration.

The condition below is true for all the values of x greater than equal to 20 (x >= 20) according to the equation (i) and (ii), where x= 1, 2, 3, 4 …

$g(x) > f(x)$ …..(iii)

Table 1: Analysis of Functions

N	Grid Searching (Proposed Algorithm) Function	Sequential Searching Function
0	5	1
3	60	13
6	117	43
9	176	91
12	237	157
15	300	241
18	365	343
21	432	463
24	501	601
27	572	757
30	645	931
33	720	1123
36	797	1333
39	876	1561
42	957	1807
…	…	…

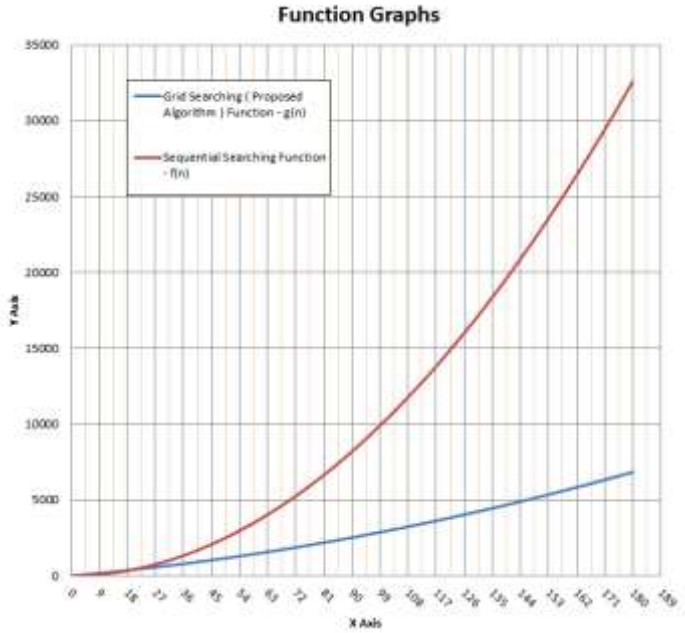


Figure 1

From the above graph (Fig. 1) we may come to a conclusion that Grid Searching function (*g(x)*) is highly effective with respect to Linear/Sequential searching (*f(x)*) when x >= 20 where x=1, 2, 3, 4, ...

5.2 Analysis of Worst-case iteration complexity

Worst-case iteration complexity measures the resources (e.g. running time, memory) that an algorithm requires in the worst-case. It gives an upper bound on the resources required by the algorithm. Thus the worst-case of an algorithm is the total number of iteration(s) required for searching an element which is not present in the 2D array/matrix, thus shows the total no. of iteration required for fulfilling a NOT FOUND condition.

NOT FOUND condition is checked for the order of matrices (n X n) where n = 3, 6, 9, 12…

According to equations (i) and (ii) from Section 5.1

$f(x) = x^2 + x + 1$ …..(i) (refer Section 5.1)

Function of Linear/Sequential searching is *f(x)*

And,

$g(x) = x^2/9 + 18x + 5$ …..(ii) (refer Section 5.1)

Function of Grid searching is *g(x)*

The highest degree of both the equations is as follows:

$f(x) = g(x) = x^2$ ….. (iii)

Thus from equation (iii) we can conclude,

The Orders of Complexity of both the equations are $O(n^2)$

Therefore,

Orders of complexity of both the algorithms are

$O(Grid\ Searching) = O(Sequential\ Searching) = O(n^2)$

But,

According to the equation (i) and (ii) considering the highest degree of the equation with constants,

We can say,

$x^2 >> x^2/9$ ….. (iv)

x=1, 2, 3, 4 …

(x^2 is much greater than $x^2/9$)

Precisely, we can also say

$O(Grid\ Searching) = O(n^2/9)$

$O(Sequential\ Searching) = O(n^2)$

Finally,

Grid searching has complexity of $O(n^2/9)$

And Sequential searching has a complexity of $O(n^2)$.

The condition below is true for all the values of n where n = 1, 2, 3, 4…

$O(n^2/9) << O(n^2)$

Table 2: Iteration Complexity

n	Order	Grid Searching (Proposed Algorithm)	Sequential Searching
1	3	1	9
2	6	4	36
3	9	9	81
4	12	16	144
5	15	25	225
6	18	36	324
7	21	49	441
8	24	64	576
9	27	81	729
10	30	100	900
11	33	121	1089
12	36	144	1296
…	…	…	…

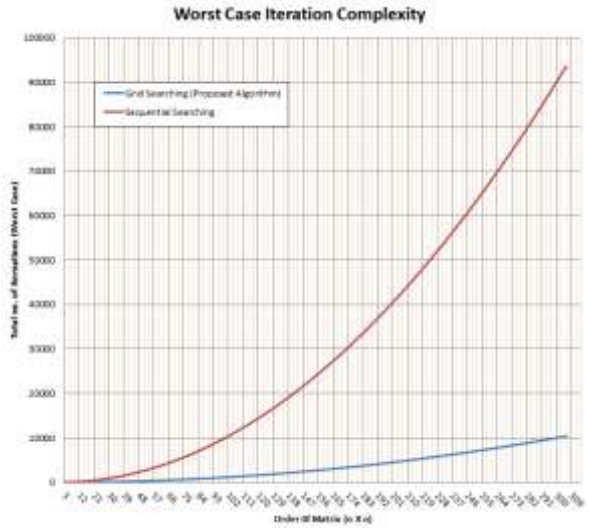


Figure 2

From the above graph (Fig. 2) we may conclude that Grid Searching is highly efficient with an increase in the order of matrix with respect to Linear/Sequential searching.

5.3 Analysis of Worst-case Time complexity

The worst-case time complexity indicates the longest running time performed by an algorithm given *any* input of size *n* (if the algorithm satisfies NOT FOUND condition then that is the longest running time of the algorithm), and thus this guarantees that the algorithm finishes on time and gives us the total worst-case time complexity.

Moreover, the order of growth of the worst-case complexity is used to compare the efficiency of two algorithms.

All the readings shown below are an average of 10 reading of a single NOT FOUND condition. Time is represented in *msec*, and NOT FOUND condition is checked for the order of matrices (n X n) where n = 3, 6, 9, 12…

Table 3: Time Complexity

N	Order	Grid Searching (Proposed Algorithm)	Sequential Searching
1	3	0.31287	0.32967
2	6	0.384615	0.494505
3	9	0.32967	0.384615
4	12	0.32967	0.43956
5	15	0.384615	0.49456
6	18	0.494505	0.512563
7	21	0.43956	0.549451
8	24	0.494505	0.549451
9	27	0.549451	0.494505
10	30	0.494505	0.494505
11	33	0.43956	0.494505
12	36	0.32967	0.549451
…	…	…	…

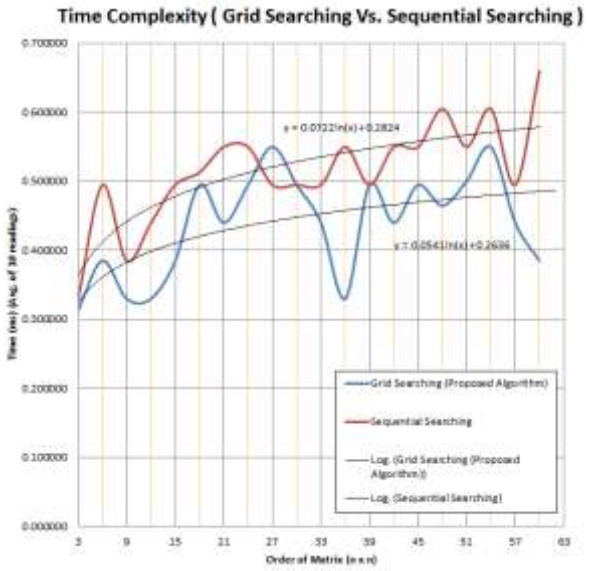


Figure 3

From the graph (Fig. 3) we obtain the logarithm values of the trend-line which is automatically generated by the Graph Analyzer from Table 3.

The logarithm trend-line for linear/sequential searching *f(x)* is as follows:-

$f(x) = 0.0722 \ln(x) + 0.2824$ .....(i)

The logarithm trend-line for Grid searching *g(x)* is as follows:-

$g(x) = 0.0541 \ln(x) + 0.2636$ …..(ii)

The condition below is true for all values of x (1, 2, 3…) according to the equation (i) and (ii),

$g(x) < f(x)$ …..(iii)

Therefore, from equation (iii) we can conclude that Grid Searching is more efficient than Sequential Searching in case of Worst-case time complexity.

5.4 Analysis of Condition checking

Condition checking is the number of conditions required to find a particular element in the matrix irrespective of the presence of the element in the matrix.

All the readings shown below are obtained from a matrix of order 165x165, where all the elements are arranged in an ascending order (but for both the algorithms elements *may not* be arranged in any particular order (Ascending or Descending)) and elements are searched in a cubic function

$f(n) = n^3$ …..(i)

Where,

n = 1, 2, 3, 4… in the matrix of order 165 x 165.

Table 4: Condition Checking

n	Element	Grid Searching (Proposed Algorithm)	Sequential Searching
1	1	1	2
2	8	20	9
3	27	76	28
4	64	190	65
5	125	371	126
6	216	141	217
7	343	25	344
8	512	524	513
9	729	672	730
10	1000	988	1001
11	1331	1472	1332
12	1728	1653	1729
13	2197	2055	2198
14	2744	2687	2745
15	3375	3069	3376
…	…	…	…

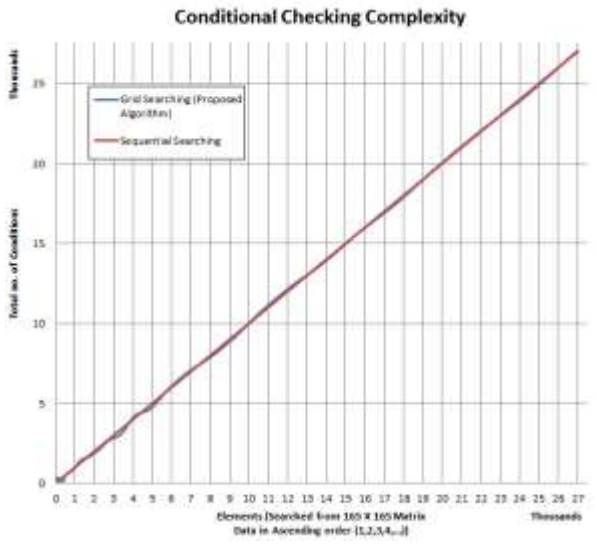


Figure 4

From the above graph (Fig. 4) we can come to a conclusion that there is not much deviation in between the two algorithms.

So, Condition checking complexity of both the algorithms is nearly same and does not affect the total time complexity.

6. CONCLUSIONS

Searching is the technique that is used to find a particular element in an array. In Grid searching and Sequential searching it is not compulsory to arrange an array in any order (Ascending or Descending) as in the case of binary search of 2D array. Result shows that Grid Searching Algorithm is working well for all input values and it is highly efficient than Sequential Searching in all of the following aspects:

1. Analysis of Functions (*f(x)* & *g(x)*) of both the algorithms
2. Worst-case iterations
3. Worst-case Time complexity of both the algorithms
4. Total no. of conditional checking

Grid searching algorithm also satisfies both the proof of correctness.

So for searching unordered set of data in square matrix Grid searching is more efficient than Sequential searching.

7. FUTURE SCOPE

Future scope of Grid searching Algorithm is as follows:-

1. If it can be implemented for any order of matrix (non-square matrix) with unordered set of data present in it.
2. Rasterization algorithm, Electro-optical display algorithm, Pixel tracing algorithm are some examples of algorithms which are derived from sequential searching and are widely used in different fields, but if the same algorithm can be built using Grid searching then the performance might increase.

8. REFERENCES

[1] E. Horowitz and S. Sahni, fundamental of Data Structure Rockville, MD: Computer Science Press, 1982.

[2] Knuth, D.E. The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison Wesley (1997), 3rd ed., 396-408.

[3] Alfred V., Aho J., Horroroft, Jeffrey D.U. (2002) Data Structures and Algorithms.

[4] Frank M.C. (2004) Data Abstraction and Problem Solving with C++. US: Pearson Education, Inc.

[5] Cormen T.H., Leiserson C.E., Rivest R.L. and Stein C. (2003) Introduction to Algorithms MIT Press, Cambridge, MA, 2nd edition.

[6] Seymour Lipschutz (2009) Data Structure with C, Schaum Series, Tata McGraw-Hill Education.

9. ABOUT THE AUTHOR

Rehan Guha is a student of Computer Application. He has already invented offline security software "Data Security- Multi-layer Folder Lock with Hiding"[5]. His other two major works include "Biometric Ticketing System"[6] and "Biometric Voting Machine and System".

[5] Published in Patent journal issue no. 44/2014 on 31/10/14. [6] Issue no. 39/2015 on 25/09/15.

All works are under process for Patent by **Intellectual Property India**.