In the Name of God

Journal of
Information Systems & Telecommunication
Vol. 1, No. 1, January-March 2013

Research Institute for ICT, Iranian Association of ICT

Affiliated to: Academic Center for Education, Culture and Research (ACECR)

Publisher: Iran University Press
ISSN: 2322-1437
eISSN: 2345-2773
Publication License: 91/13216

Editorial office Address: No.5, Saeedi Alley, Kalej Intersection., Enghelab Ave., Tehran, Iran,
P.O.Box: 13145-799 Tel: (+9821) 88930150 Fax: (+9821) 88930157
Email: info@jist.ir
URL: www.jist.ir

Indexed in:

- Journal of Information Systems & Telecommunication www.jist.ir
- Scientific Information Database (SID) www.sid.ir
- Islamic World Science Citation Center (ISC) www.isc.gov.ir
- Regional Information Center for Sciences and Technology (RICeST) www.srlst.com
- Magiran www.magiran.com

This Journal is published with scientific supported by the Advanced Information systems (AIS) and Information Technology Business Model (ITBM) research groups in Research Institute for ICT.

Table of Content

A New Upper Bound for Free Space Optical Channel Capacity Using a Simple Mathematical in Equality

Arezu Rezazadeh*
Department of Electrical Engineering, Ferdowsi University of Mashhad
arezu.rezazadeh@gmail.com
Ghosheh Abed Hodtani
Department of Electrical Engineering, Ferdowsi University of Mashhad
ghodtani@gmail.com

Received: 05/Aug/2012 Accepted: 15/Dec/2012

Abstract
In this paper, by using a simple mathematical inequality, we derive a $ new upper bound fkr the capacity of$ free space optical channel in coherent case. Then, by applying general fading distribution, we obtain an upper bound for mutual information in non-coherent case. Finally, we derive the corresponding optimal input distributions for both coherent and non-coherent cases, compare the results with previous works numerically and illustrate that our results subsume some of previous results in special cases.

Keywords: Mathematical Inequality, Capacity, Upper Bound, Free Space Optical Channel, Optimal Input Distributions.

1. Introduction

Free space optical (FSO) channel is important because of high transmission rate, power efficiency, high bandwidth and its safety.

To design communication link with high performance, it is necessary to study its properties from information theoretical viewpoint. To determine channel capacity, optimum input distribution should be obtained. By considering input constraints, the optimum input distribution is derived. In FSO channel, for eye safely and physical limitations, average and peak power constraints are imposed on transmitted signal [1]. The mathematical representation for FSO channel is [1]:

$$Y = HX + Z, \tag{1}$$

Where, X is the channel input, *Y* is the output and *Z* is the Gaussian noise with zero mean and variance of *σ2* or *Z ~ N(0;σ2)*. H represents channel fading which has the probability density function *f(h)*. The input constraints are [1]:

$$0 \le X \le A, \quad E\{X\} \le P, \quad \rho = \frac{A}{P}, \tag{2}$$

Where A is the peak-amplitude limit and *P* is the average power limit and ρ is the ratio of optical peak to average power.

Part of this paper was accepted at Australian Communication Theory Workshop 2012 (AusCTW2012).

The authors are with the Department of Electrical Engineering, Ferdowsi University of Mashhad, Mashhad, Iran (e-mail: arezu.rezazadeh@gmail.com; ghodtani@gmail.com).

Previous Works: In [2] with constraints on input amplitude and power, it was shown that in coherent receiver, the capacity-achieving input distribution is discrete with a finite number of mass points. In other words, the input maximizing *I(X; Y| h)* is:

$$P = \{p_x(x) : p_x(x) = \sum_{i=0}^{K} a_i \delta(x - x_i), x_i \in [\circ, A],$$
$$a_i \ge 0, \sum_{i=\circ}^{K} a_i = 1, K \in Z^+, P \ge \sum_{i=\circ}^{K} x_i a_i \}, \tag{3}$$

Where *δ(x)* is the delta function and *Z+* is the set of positive integers. The number of mass points is *K + 1*, where ai and *xi* are the amplitudes and positions of the ith mass points, respectively [1], [2].

In [1] instead of maximizing mutual information, source entropy is maximized for the capacity of FSO channel.

In [3] under non-negativity and average optical power constraints lower and upper bounds for *I(X; Y|h = 1)* are derived. The lower and upper bounds are derived by maximizing source entropy and using a sphere packing argument respectively.

In [4] bounds for *I(X;Y|h = 1)* are derived by using a dual minimax problem (instead of maximizing the mutual information over distributions on the channel input alphabet, average relative entropy is minimized over

* Corresponding Author

distributions on the channel output alphabet). At high-power regime, a lower bound for *I(X; Y|h = 1)* is also proposed by using the entropy power inequality.

In [5] by considering Gaussian maximum entropy for *H(Y|h)*, an upper bound for *I(X; Y|h)* has been derived. Then by averaging over the gamma-gamma atmospheric turbulence for h, an upper bound for *I(X;Y)* (non-coherent case) has been determined.

Our Work: In this paper, we derive a new upper bound for the capacity of FSO channel in both coherent and non-coherent cases and determine the corresponding optimum input distributions for these two cases.

As pointed before, for additive noise with input peak and power constraints, the optimum input distribution is discrete with finite number of mass points [2]. Similarly for coherent case with these constraints, capacity achieving distribution is discrete with finite number of mass points. By considering this fact and using simple mathematical inequality, we determine a new upper bound for capacity of FSO channel. Then we extend the result to the non-coherent case with arbitrary *f(h)* and finally we determine the corresponding input distribution and compare the results with previous works.

Paper Organization: This paper has four sections. In section II, an upper bound for *I(X;Y|h)* and the corresponding input distribution is found. In section III, an upper bound for *I(X; Y)* (non coherent case) is derived by averaging over distribution of *f(h)*. Then we will maximize the upper bound of *I(X;Y)* over all input distributions. The paper concludes in section IV.

2. An Upper Bound for I(X; Y|h) and the Corresponding Input Distribution

In this section, first we determine an upper bound for *I(X;Y|h)* and then determine the corresponding input distribution. For discrete-time Gaussian channels [6], capacity can be expressed as:

$$C = \max_{f_x(x)} I(X;Y) = \max_{f_x(x)} \int I(X;Y|h) f(h) dh, \quad (4)$$

To reach *I(X;Y)*, we simplify *I(X;Y|h)*. *X* and *H* are independent, thus the mutual information, between channel input and output is [1]:

$$I(X;Y|H=h) = H(Y|H=h) - H(Y|X,H=h)$$
$$= -\int f(y|h)\log_2 f(y|h)dy + \int f(y|h,x)\log_2 f(y|h,x)dy \quad (5)$$

Where, in view of (1):

$$(y|h,x) \sim N(hx,\sigma^2) \quad (6)$$

$$f(y|h) = \int f_x(x) f(y|h,x)dx \quad (7)$$

Where *N(μ,σ2)* denotes a Gaussian distribution with mean μ and variance *σ2* and *fx(x)* is the input distribution in (3). Therefore,

$$f(y|h) = \int f_X(x) f(y|h,x)dx$$
$$= \int \sum_{i=0}^{K} a_i \delta(x - x_i) \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx)^2}{2\sigma^2}} dx$$
$$= \sum_{i=0}^{K} a_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_i)^2}{2\sigma^2}}.$$

So,

$$I(X;Y|h) = -\int f(y|h)\log_2(f(y|h))dy - H(z) = -\int \sum_{i=0}^{K} a_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_i)^2}{2\sigma^2}} \log_2\left(\sum_{j=0}^{K} a_j \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_j)^2}{2\sigma^2}}\right)dy - H(z). \quad (8)$$

Since the above integral cannot be evaluated analytically, we will determine an upper bound for *I(X;Y|h)*.

A. Upper Bound for I(X;Y|h)

In order to find an upper bound for *I(X;Y|h)*, we write *I(X;Y|h)* in terms of ais. From (8) we have:

$$I(X;Y|H=h) = -H(z) - \int \sum_{i=0}^{K} a_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_i)^2}{2\sigma^2}} \log_2\left(\sum_{j=0}^{K} a_j \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_j)^2}{2\sigma^2}}\right)dy$$
$$= -H(z) - (\log_2 e)\left[\int \sum_{i=0}^{K} a_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_i)^2}{2\sigma^2}} \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)dy + \int \sum_{i=0}^{K} a_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_i)^2}{2\sigma^2}} \ln\left(\sum_{j=0}^{K} a_j e^{-\frac{(y-hx_j)^2}{2\sigma^2}}\right)dy\right]$$
$$\overset{a}{=} -H(z) - \log_2\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - (\log_2 e)\left[\int \sum_{i=0}^{K} a_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_i)^2}{2\sigma^2}} \ln\left(\sum_{j=0}^{K} a_j e^{-\frac{(y-hx_j)^2}{2\sigma^2}}\right)dy\right] \quad (9)$$

where, a in (9) follows from the fact that $\sum_{i=0}^{K} a_i = 1$ and $\int \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-hx_i)^2}{2\sigma^2}} dy=1$. Because ais are less than one, so $a_i e^{-\frac{(y-hx_i)^2}{2\sigma^2}}$ is less than 1. Furthermore

$$u_i \le 1 \Rightarrow \sum u_i \ge \prod u_i \Rightarrow -\log \sum u_i \le -\log \prod u_i$$

hence,

$$-\ln(\sum_{i=0}^{K} a_i e^{-\frac{(y-hx_i)^2}{2\sigma^2}}) \le -\ln(\prod_{i=0}^{K} a_i e^{-\frac{(y-hx_i)^2}{2\sigma^2}}). \quad (10)$$

Now, we can determine an upper bound for *I(X;Y|h)*. From (9) and (10), the following upper bound is obtained.

$$I(X;Y \mid H=h) \le A_1 + A_2 h^2 \quad (11)$$

Where,

$$A_1 = -\log_2(\frac{1}{\sqrt{2\pi\sigma^2}}) - H(z) - \log_2(\prod_{j=0}^{K} a_j) + \frac{K+1}{2}\log_2^e, \quad (12)$$

And

$$A_2 = \log_2^e \sum_{n=0}^{K}\sum_{m=0}^{K} \frac{a_n}{2\sigma^2}(x_m - x_n)^2. \quad (13)$$

Now we should determine the corresponding input distribution.

B. Determining Optimum input distribution for upper bound of I(X;Y|h)

Here, we determine ais such that the upper bound in (11), regarding the constraints, becomes maximum. We use Lagrangian coefficients to determine optimum input distribution.

$$J_1 = A_1 + A_2 h^2 + \lambda_1^1(\sum a_i - 1) + \lambda_2^1(\sum a_i x_i - P). \quad (14)$$

To solve the optimization problem, considering constraints $\sum_{i=0}^{K} a_i = 1$ and $\sum_{i=0}^{K} a_i li = P$, we make $\frac{\partial J_1}{\partial a_i} = 0$. For subset of input distribution with *K + 1* equally spaced mass points i.e., *xi = il*, where, $l = \frac{A}{K}$, we have:

$$a_i = \frac{1}{B_1 + B_2} \quad (15)$$

Where,

$$B_1 = (K+1) + \frac{h^2}{2\sigma^2} lK(K+1)P + \frac{h^2}{2\sigma^2}\left\{-(K+1)\sum_{j=0}^{K} a_j j^2 l^2 - K(K+1)l^2 i + i^2 l^2 (K+1)\right\},$$

and

$$B_2 = \frac{(il - P)}{(\sum_{j=0}^{K} a_j (jl)^2 - P^2)}\left[l\left(\frac{K(K+1)}{2}\right) - (K+1)P - \frac{h^2}{2\sigma^2}\left\{ lK(K+1)P^2 + (K+1)\sum_{j=0}^{K} a_j (lj)^3 - \left[(K+1)P + K(K+1)l\right]\sum_{j=0}^{K} a_j j^2 l^2 \right\}\right].$$

Optimum input distribution which maximizes the upper bound in (11) is derived via (15). It is clear that (15) is non linear and should be determined numerically. In general, optimal input distributions are different for each A (peak amplitude limit), *σ2* (variance of noise) and h. So for a given *A/σ*, h and ρ, optimal input distribution is determined numerically. By considering *h = 1*, *A = 1* and *σ=1* amplitude of mass points for, *ρ= 10* and *ρ = 2.5* are presented in Tables I and II respectively.

In coherent case by applying *h = 1*, to (15) we compare our derived upper bound with bounds which are derived in [4]. For a given *A/σ* and *ρ*, amplitude of mass points are computed for several K (number of mass points), and the corresponding upper bounds, which are derived from (11), are collected in a collection. The optimum number of mass points correspond to the upper bound which has minimum distance with lower bound. Fig. 1 illustrates the comparison between our upper bound (11) and bounds derived in [4]. At low *A/σ*, our upper bound is showing tighter performance than upper bounds which are proposed in [4]. Although at high *A/σ* there is a great gap between upper bounds derived from (11) and lower bound derived in [4], but our proposed upper bound is determined simply. The coherence time, for FSO channel is on the order of 1-100 msec [1]. To plot figure, we consider the coherence time 1 msec.

TABLE I: OPTIMAL INPUT DISTRIBUTION FOR COHERENT CASE

(15), WHEN h = 1, ρ= 10 AND A/σ = 0dB

Number of mass points	a0	a1	a2	a3	a4
K=1	0.9	0.1			
K=2	0.8505	0.0989	0.0505		
K=3	0.8181	0.0975	0.0507	0.0337	
K=4	0.794	0.0964	0.0505	0.0339	0.0253

TABLE II: OPTIMAL INPUT DISTRIBUTION FOR COHERENT CASE

(15),WHEN h = 1,ρ= 2.5 AND A/σ = 0dB

Number of mass points	a0	a1	a2	a3	a4
K=1	0.6	0.4			
K=2	0.4307	0.3386	0.2307		
K=3	0.3409	0.2808	0.2158	0.1626	
K=4	0.2836	0.2398	0.1950	0.1563	0.1253

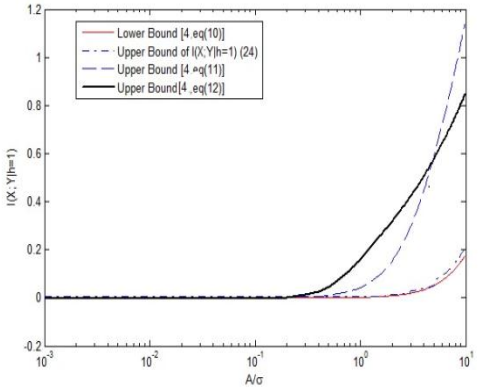


Fig. 1. Comparison of upper and lower bounds at low A/σ when h = 1 and ρ= 10.

3. An Upper bound for I(X; Y) and the corresponding input distribution

We want to compute $I(X;Y) = \int I(X;Y|h) f(h) dh$ and then maximize I(X;Y) over all input distributions. First we describe *f(h)* in terms of hyper-geometric functions and then continue aiming at finding the upper bound.

Description of *f(h)* in Terms of Hyper-geometric Functions

In FSO channel, the channel state *h* is the product of $g_a\, h_a\, h_p$, where g_a is the deterministic path loss, h_a is the random attenuation due to atmospheric turbulence and well modeled by a Gamma-Gamma distribution, and hp is the random attenuation due to geometric spread and pointing errors [1], [7], [8]. The probability density of *h* i.e., *f(h)* in [1] and [7] is expressed as:

$$f_h(h) = \frac{\gamma^2 h^{\gamma^2-1}}{(A_0 g_a)^{\gamma^2}} \int_{h/A_0 g_a}^{\infty} h_a^{-\gamma^2} f_{ha}(h_a) dh_a, \tag{16}$$

Where,

$$f_{h_a}(h_a) = \frac{2(\alpha\beta)^{\frac{\alpha+\beta}{2}}}{\Gamma(\alpha)\Gamma(\beta)} (h_a)^{\frac{\alpha+\beta}{2}-1} K_{\alpha-\beta}(2\sqrt{\alpha\beta h_a}), \tag{17}$$

Where Kα-*β(.)* is the modified Bessel function of the second kind, Γ (·) is the gamma function, and 1/*α* and 1/*β* are the variances of small and large scale eddies respectively [1], and an expression for ga, *γ* and A0 is given in [1], [7]. A closed form for probability density function of *h* in terms of hyper-geometric functions, was computed in [8]. Considering $\Gamma(s)\Gamma(1-s) = \pi \csc(\pi s)$, the probability density of *h* [8,eq. (13)] can be expressed as:

$$f_h(h) = \frac{\gamma^2 h^{-1}}{\Gamma(\alpha)\Gamma(\beta)} \left\{ \left(\frac{\alpha\beta h}{A_0 g_a}\right)^{\gamma^2} \Gamma(\beta-\gamma^2)\Gamma(\alpha-\gamma^2) \right.$$
$$+ \frac{\left(\frac{\alpha\beta h}{A_0 g_a}\right)^{\alpha} \Gamma(\beta-\alpha)}{(\gamma^2-\alpha)} \times$$
$${}_1F_2\left(\alpha-\gamma^2; \alpha-\beta+1, \alpha-\gamma^2+1; \frac{\alpha\beta h}{A_0 g_a}\right)$$
$$+ \frac{\left(\frac{\alpha\beta h}{A_0 g_a}\right)^{\beta} \Gamma(\alpha-\beta)}{(\gamma^2-\beta)} \times$$
$$\left. {}_1F_2\left(\beta-\gamma^2; \beta-\alpha+1, \beta-\gamma^2+1; \frac{\alpha\beta h}{A_0 g_a}\right) \right\} \tag{18}$$

Where 1F2(a;b,c;z) is a generalized hyper-geometric function with series representation:

$${}_1F_2(a;b,c;z) = \sum_{k=0}^{\infty} \frac{(a)_k}{(b)_k (c)_k} \frac{z^k}{k!}$$

Here (.)k represents the Pochhammer symbol, which is defined by

$(z)_0 = 1$ and $(z)_n = z(z+1)(z+2)\ldots(z+n-1) = \Gamma(z+n)/\Gamma(z)$.

We expressed *f(h)*. Now, we can determine an upper bound for *I(X;Y)*.

Maximizing the Mutual Information and Determining an Expression for the Input Distribution

First we compute the following expression, then we determine ais, (with considering constraints) such that the upper bound of *I(X;Y)* will be maximized.

$$I(X;Y) \le \int (A_1 + A_2 h^2) f(h) dh. \tag{19}$$

We know that [9]:

$$\int_0^{K_c} h^a \; {}_1F_2(a_1; a_2, a_3; bh) dh =$$
$$= \frac{K_c^{a+1}}{a+1} \; {}_2F_3(a_1, a+1; a_2, a_3, a+2; bK_c) \tag{20}$$

Where,

$$ {}_2F_3(a_1,a_2;b_1,b_2,b_3;z)=\sum_{k=0}^{\infty}\frac{(a_1)_k\,(a_2)_k}{(b_1)_k\,(b_2)_k\,(b_3)_k}\frac{z^k}{k!} \quad (21)$$

So we can write:

$$I(X;Y)\le\int (A_1+A_2h^2)f(h)dh=$$

$$=\int_0^{K_c}A_1f(h)dh+\int_0^{K_c}A_2h^2f(h)dh.$$

Notice that, the integral of expectation, is defined from zero to constant K_c. It means that

$$0\le h\le K_c$$

We will see the dependence of mass points on this parameter (K_c) later. Considering (18) and (20), the upper bound for I(X; Y) can be expressed as:

$$I(X;Y)\le A_1I_1+A_2I_2 \quad (22)$$

where

$$I_1=\frac{\gamma^2}{\Gamma(\alpha)\Gamma(\beta)}[I_{1a}+I_{1b}+I_{1c}] \quad (23)$$

And

$$I_2=\frac{\gamma^2}{\Gamma(\alpha)\Gamma(\beta)}[I_{2a}+I_{2b}+I_{2c}] \quad (24)$$

Where,

$$I_{1a}=(\frac{\alpha\beta}{A_0g_a})^{\gamma^2}\Gamma(\beta-\gamma^2)\Gamma(\alpha-\gamma^2)\frac{K_c^{\gamma^2}}{\gamma^2}$$

$$I_{1b}=\frac{(\frac{\alpha\beta}{A_0g_a})^{\beta}\Gamma(\alpha-\beta)}{(\gamma^2-\beta)}\times$$

$$\frac{K_c^{\beta}\ {}_2F_3(\beta-\gamma^2,\beta;\beta-\alpha+1,\beta-\gamma^2+1,\beta+1;\frac{\alpha\beta}{A_0g_a}K_c)}{\beta}$$

$$I_{1c}=\frac{(\frac{\alpha\beta}{A_0g_a})^{\alpha}\Gamma(\beta-\alpha)}{(\gamma^2-\alpha)}\times$$

$$\frac{K_c^{\alpha}\ {}_2F_3(\alpha-\gamma^2,\alpha;\alpha-\beta+1,\alpha-\gamma^2+1,\alpha+1;\frac{\alpha\beta K_c}{A_0g_a})}{\alpha}$$

And

$$I_{2a}=(\frac{\alpha\beta}{A_0g_a})^{\gamma^2}\Gamma(\beta-\gamma^2)\Gamma(\alpha-\gamma^2)\frac{K_c^{\gamma^2+2}}{\gamma^2+2}$$

$$I_{2b}=\frac{(\frac{\alpha\beta}{A_0g_a})^{\beta}\Gamma(\alpha-\beta)}{(\gamma^2-\beta)}\times$$

$$\frac{K_c^{\beta+2}\ {}_2F_3(\beta-\gamma^2,\beta+2;\beta-\alpha+1,\beta-\gamma^2+1,\beta+3;\frac{\alpha\beta K_c}{A_0g_a})}{\beta+2}$$

$$I_{2c}=\frac{(\frac{\alpha\beta}{A_0g_a})^{\alpha}\Gamma(\beta-\alpha)}{(\gamma^2-\alpha)}\times$$

$$\frac{K_c^{\alpha+2}\ {}_2F_3(\alpha-\gamma^2,\alpha+2;\alpha-\beta+1,\alpha-\gamma^2+1,\alpha+3;\frac{\alpha\beta K_c}{A_0g_a})}{\alpha+2}$$

and 2F3 has been defined in (21). Now, we maximize the upper bound of I(X; Y) over all input distributions and derive an expression for the input.

Determining Optimal input Distribution which Maximizes Our Upper Bound of *I(X; Y)*

We should determine a_is such that the upper bound in (22), regarding the constraints, becomes maximum. We define *J* as the Lagrangian associated with the optimization problem. Again similar to previous section, to solve the optimization problem, considering constraints $\sum_{i=0}^{K}a_i=1$ and $\sum_{i=0}^{K}a_i li=P$, we make $\frac{\partial J}{\partial a_i}=0$. For subset of input distribution with *K + 1* equally spaced mass points i.e., *xi = il*, where, $l=\frac{A}{K}$ we have:

$$J=A_1I_1+A_2I_2+\lambda_1(\sum a_i-1)+\lambda_2(\sum a_i li-P).$$

by considering constraints $\sum_{i=0}^{K}a_i=1$ and $\sum_{i=0}^{K}a_i li=P$, the optimized *ais* which maximize the upper bound of *I(X; Y)*, can be expressed as:

$$a_i=\frac{1}{D_1+D_2} \quad (25)$$

Where,

$$D_1=(K+1)I_1+\frac{I_2}{2\sigma^2}lK(K+1)P$$

$$+\frac{I_2}{2\sigma^2}\left(-(K+1)\sum_{j=0}^{K}a_j j^2l^2-K(K+1)l^2i+i^2l^2(K+1)\right),$$

and

$$D_2 = \frac{(il - P)}{(\sum_{j=0}^{K} a_j (jl)^2 - P^2)}\left[I_1 l \left(\frac{K(K+1)}{2} \right) - (K+1) I_1 P \right.$$

$$\left. - \frac{I_2}{2\sigma^2} \left\{ lK(K+1)P^2 + (K+1) \sum_{j=0}^{K} a_j (lj)^3 \right. \right.$$

$$\left. \left. - \left[(K+1)P + K(K+1)l \right] \sum_{j=0}^{K} a_j j^2 l^2 \right] \right\}.$$

So, the optimal input distribution, which maximizes the upper bound of *I(X; Y)*, has the above relation.

It is clear that, (25) is non linear and depends on channel parameters. Notice that neither the correct

Number of mass points (K) nor the values of them

(a_i) are known. The equation (25) is non linear and depends on the channel parameters. We should determine channel parameters, to compute a_is. But due to complexity of equation (25), the numerical

Calculation have been done just for K = 1. It can be seen easily that, when K = 1, a_is just depend on ρ. So it is clear that they are independent on Kc, which is the upper limit of integral in computing expectation of *I(X; Y |h)*, and other channel parameters. Thus, for K = 1, there is no need to know channel parameters. When K = 1, a_is are determined as a function of ρ. By using (25) and (2), we have:

$$K = 1 \Rightarrow P = \sum_{i=0}^{1} a_i li = a_1 l$$

$$\Rightarrow a_1 = \frac{1}{\rho}, \qquad a_0 = \frac{\rho - 1}{\rho} \tag{26}$$

It is the exact result, given in [1]. Farid and Hranilovic have shown that, for K = 1, the amplitude of mass points are given by the following

Equation [1]:

$$[P_0, P_1] = [\frac{\rho - 1}{\rho}, \frac{1}{\rho}].$$

For K = 1, the amplitude of mass points, for coherent and non coherent, are the same and determined from (26).

4. Conclusion

In this paper, by using a simple mathematical inequality, we determined new upper bounds for capacity of FSO channel in coherent and non coherent cases. For h = 1 we compare our results with previous works. At low SNR our upper bound shows tighter performance. For non coherent case, optimum input distribution depends on channel parameters, but for two mass points, optimum value of mass points are independent of channel parameters and just depend on ρ. Our results subsume some of the previous ones in special cases.

References

[1] A. Farid and S. Hranilovic, "Channel capacity and non uniform signalling for free-space optical intensity channels," *Selected Areas in Communications, IEEE Journal on*, vol. 27, no. 9, pp. 1553 –1563, December 2009.

[2] J. G. and Smith, "The information capacity of amplitude and variance constrained scalar Gaussian channels," *Information and Control*, vol. 18, no. 3, pp. 203 – 219, 1971.

[3] A. Farid and S. Hranilovic, "Capacity bounds for wireless optical intensity channels with Gaussian noise," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 6066 –6077, dec. 2010.

[4] A. Lapidoth, S. Moser, and M. Wigger, "On the capacity of free-space optical intensity channels," *in Proc.IEEE Int. Symp. Information Theory*, July 2008, pp. 2419 –2423.

[5] A. Garca-Zambrana, C. Castillo-Vzquez, and B. Castillo-Vzquez, "On the capacity of fso links over gamma-gamma atmospheric turbulence channels using ook signaling," *EURASIP Journal on Wireless Communications and Networking*, 2010.

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. John Wiley & Sons, 2006.

[7] A. Farid and S. Hranilovic, "Outage capacity optimization for free-space optical links with pointing errors," *IEEE J. Lightwave Tech*, vol. 25, no. 7, pp. 1702 –1710, July 2007.

[8] H. Sandalidis, T. Tsiftsis, and G. Karagiannidis, "Optical wireless communications with heterodyne detection over turbulence channels with pointing errors," *Journal of Lightwave Technology*, vol. 27, no. 20, pp. 4440 – 4445, Oct.15, 2009.

[9] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, 7th ed. Academic Press, 2007.

Achieving Better Performance of S-MMA Algorithm in the OFDM Modulation

Saeed Ghazi-Maghrebi*
Islamic Azad University, Share-e-Rey Branch, Iran
ghazimaghrebi@ieee.org
Babak Haji Bagher Naeeni
IRIB University, Tehran, Iran
bnaeeni@yahoo.com
Mojtaba Lotfizad
Department of Electrical & Computer Engineering, Tarbiat Modares University, Tehran, Iran
lotfizad@yahoo.com

Received: 12/Sep/2012 Accepted: 11/Mar/2013

Abstract
Effective algorithms in modern digital communication systems provide a fundamental basis for increasing the efficiency of the application networks which are in many cases neither optimized nor very close to their practical limits. Equalizations are one of the preferred methods for increasing the efficiency of application systems such as orthogonal frequency division multiplexing (OFDM). In this paper, we study the possibility of improving the OFDM modulation employing sliced multi-modulus algorithm (S-MMA) equalization. We compare applying the least mean square (LMS), multi modulus algorithm (MMA) and S-MMA equalizations to the per tone equalization in the OFDM modulation. The paper contribution lies in using the S-MMA technique, for weight adaptation, to decreasing the BER in the OFDM multicarrier modulation. For more efficiency, it is assumed that the channel impulse response is longer than the cyclic prefix (CP) length and as a result, the system will be more efficient but at the expense of the high intersymbol interference (ISI) impairment existing. Both analysis and simulations demonstrate better performance of the S-MMA compared to LMS and MMA algorithms, in standard channels with additive white Gaussian noise (AWGN) and ISI impairment simultanously. Therefore, the S-MMA equalization is a good choice for high speed and real-time applications such as OFDM based systems.

Keywords: Cyclic Prefix, Equalization, ISI, LMS, MMA, OFDM, SMMA.

1. Introduction

During recent years, the authors have designed different equalizations for different modulation schemes [1]. Achieving more efficiently orthogonal frequency division multiplexing (OFDM) performance, only by changing the equalization, is the main idea of this paper.

In the most digital communication systems, the inter symbol interference (ISI) occurs due to band-limited channels or multipath propagation. The channel equalization is one of the techniques to decreasing the effect of the ISI [2]. Another way for the cost effective handling of the ISI comes at the expense of the bandwidth efficiency reduction caused by inserting the CP. It is apparent that for more efficiently, the OFDM modulation that can perform well at short CP length is highly desired [3].

In recent years, because of the severity of distortion, the problem of alleviating insufficient-CP length distortion has received a great deal of attention. Following the early work in [4], where the authors have shorten the channel to reduce the complexity of maximum likelihood sequence estimation (MLSE), the authors in [5] propose a time domain equalizer (TEQ) for digital subscriber line (DSL) systems. In [6], the insufficient-CP distortion was eliminated by a precoder at the transmitter. Moreover, the precoder essentially performs a matrix inversion and thus is prohibitively complex. The work in [6] did not fully take into account the inherent receiver noise and the transmitter power constraint. For some channels, this precoder will result in increasing transmitter power budget and scaling down the precoder coefficients which causes a significant data rate loss. The complexity is then significantly reduced as reported in [5] but the implementation is only applicable for systems with zero CP and still suffers from the power increment problem [7].

* Corresponding Author

The purpose of an equalization algorithm, in single carrier systms, is to make the equalizer match to the inverse of the communication channel impulse response, thus opening the eye of the communication system and allowing for a correct retrieval of the transmitted symbols [8]. However, in the multicarrier systems a TEQ is usd in transmitter for shortening the channel impulse response length and also a per-tone equalizer is used in the receiver for decreasing the ISI. There are many different algorithms for updating the tap values of equalizers. The constant modulus algorithm (CMA) [9, 10] is one of these algorithms that have been used for quadrature amplitude modulation (QAM) signals. In order to improve its performance, the authors have been proposed the multi-modulus algorithm (MMA) [8]. In this work, we propose a new MMA algorithm i.e. sliced multi-modulus algorithm (S-MMA) equalization [8], for updating per-tone equalization [11] taps in the OFDM multicarrier modulation. For more qualifying the proposed S-MMA algorithm performance, we test the algorithm with Stanford University Interim (SUI) standard channels in the presence of the ISI, due to an insufficient CP length, and additive white Gaussian noise (AWGN) simultaneously.

The paper is organized as follows. The OFDM modulation description is explained in Section 2 and analysis of the per-tone equalization in the OFDM modulation is explained in Section 3 and analysis of the CP insertion in OFDM modulation is described in Section 4 and S-MMA equalization performance analysis is described in Section 5. In Sections 6 and 7, simulation results and conclusions are presented respectively.

2. OFDM modulation description

The basic idea of the OFDM is spliting up a high rate data stream into a number of parallel lower rate data sub-streams, which are transmitted simultaneously over different sub-carriers [13]. The OFDM modulation is resistant to multipath interference and frequency selective fading. The OFDM systems also have a relatively simple receiver structure compared to single carrier transmission in frequency selective fading channels [4]. However, the OFDM utilizes the spectrum much more efficiently by spacing the channels much closer together [14]. The OFDM has good performances of anti-ISI, anti-decline, resisting interfere of narrow-band, fitting for asymmetrical transmission and robustness to multipath fading [15]. Because of these advantages, the OFDM has been adopted in both wireless and wired applications in recent years [16].

In the block diagram of an OFDM transmitter, as shown in Fig. 1, a sampled analog signal passes through an analog to digital (A/D) converter and then the resulting bitstream is divided into a number of parallel blocks with a serial to parallel (S/P) converter. These blocks are the input of the constellation mapper, which is basically representing segments of bits as spectral coefficients.

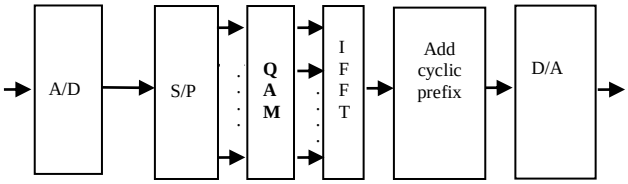


Fig.1. OFDM transmitter block diagram

The resulting sub-channels are orthogonal to each other as long as the CP is longer than the channel impulse response. Otherwise, the system will suffer from insufficient-CP length distortion, which is composed of inter carrier interference (ICI) and the ISI [7, 16].

The channel is modeled so that it adds two forms of interference. i.e. ISI and ICI impairments. It is illustrated that not only ISI but also ICI is caused by the collapse of orthogonality in the received signal. As a result, both the channel identification and equalization become difficult, and the communication performance cannot be guaranteed [17]. The ISI and ICI impairments is removed by turning the linear convolution into a cyclic convolution via insertion a CP at the beginning of each input data stream blocks [18] - [19].

In the receiver, as shown in Fig. 2, the received signal is again broken up into parallel blocks. The CPs are removed and then the FFT of each block is calculated. The equalizer attempts to reduce the ISI in the received signal and maximizes the SNR at the input of the decision circuit. A constellation demapper converts the complex values to a bit stream.

Due to the additive noise, the received constellation points deviate from their location in the original constellation. For recovering the received bitstream, a nearest-neighbor approximation method is computed at each point. The blocks of bits are concatenated back into a single bitstream and then undergoes a D/A convertor and finally back to a sampled analog signal.

3. Analysis of the per tone equalization in the OFDM modulation

With the aid of inverse fast Fourier transform (IFFT) algorithm and appending a CP between the individual blocks at the transmitter and using the fast Fourier transform (FFT) algorithm at the receiver, a broadband frequency-selective channel is converted into a set of parallel flat fading sub-channels or tones [15].

Multicarrier modulation is a powerful technique for providing broadband wireline and wireless communication to customer premises. In wireline applications, multicarrier systems are are used in discrete multitone (DMT) modulation in different variant DSL. Multicarrier is also used in wireless applications such as OFDM defined in IEEE802.11a and HIPERLAN2 standards.

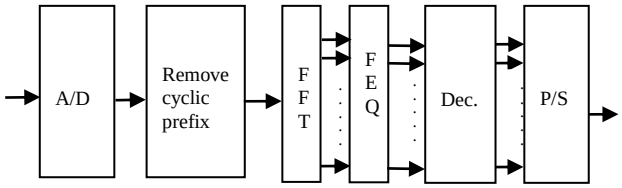


Fig.2. OFDM receiver block diagram

Practical systems use a relatively short cyclic prefix and employ equalization to compensate for the channel effects. The OFDM receiver consists of a real T-tap TEQ, as shown in Fig. 3, for shortening channel impulse response which its outputs are fed to an FFT block that is followed by a complex 1-tap frequency domain equalizer (FEQ) to compensate for the channel amplitude and phase effects. In wireless applications, the goal of TEQ is bit error rate minimization and fast adaptation to non-stationary environment are desired. Per tone equalizer is proposed in 2001 which the structure of a T-tap TEQ in combination with a complex 1-tap FEQ per tone is modified into a structure with a complex T-tap FEQ per tone. As a result, each tone is equalized separately and this leads to a higher bit rate and reduced sensitivity to the synchronization delay[16].

A crucial aspect in this process is that the channel impulse response length may be shorter or longer than the CP length. In the former, the ISI is removed and only the FEQ is required, whereas for the latter both the frequency and TEQ are needed [11]. The received symbol is the convolution of the transmitted symbol and the channel impulse response h=[h0,…, hL], plus additive noise.

For inserting the CP to each symbol, we use the P matrix as bellow

$$\mathbf{P}=\begin{bmatrix}\mathbf{0}\,|\,\mathbf{I}_c\\ \hline \mathbf{I}_N\end{bmatrix} \quad (1)$$

which I and 0 matrices are 'identity' and 'zero' matrices respectively and their indexes show the size of the matrices [11]. Considering three succssesive OFDM symbols $X_{1:N}^{(c)}$ for time $C=k-1,k,k+1$, the received signal will be [11]

$$\overbrace{\begin{bmatrix} y_{k.s+\nu-T+2}\\ \vdots\\ \vdots\\ y_{(k+1).s}\end{bmatrix}}^{\mathbf{y}}=\left[\begin{array}{c|ccc}\mathbf{O}_{(1)} & \bar{\mathbf{h}} & \mathbf{0} & \cdots & \mathbf{0}\\ & \mathbf{0} & \bar{\mathbf{h}} & \ddots & \vdots\\ & \vdots & \ddots & \ddots & \mathbf{0}\\ & \mathbf{0} & \cdots & \mathbf{0} & \bar{\mathbf{h}}\end{array}\right]\mathbf{O}_{(2)}$$

$$\begin{bmatrix}\mathbf{P} & \mathbf{O} & \mathbf{O}\\ \mathbf{O} & \mathbf{P} & \mathbf{O}\\ \mathbf{O} & \mathbf{O} & \mathbf{P}\end{bmatrix}\begin{bmatrix}I_{NFFT} & \mathbf{O} & \mathbf{O}\\ \mathbf{O} & I_{NFFT} & \mathbf{O}\\ \mathbf{O} & \mathbf{O} & I_{NFFT}\end{bmatrix}\overbrace{\begin{bmatrix}X_{1:N}^{(K-1)}\\ X_{1:N}^{(K)}\\ X_{1:N}^{(K-1)}\end{bmatrix}}^{\hat{\mathbf{x}}}$$

$$+\overbrace{\begin{bmatrix} n_{S.k+\nu-T+2}\\ \vdots\\ \mathrm{n}_{(k+1)s}\end{bmatrix}}^{\mathbf{n}}$$

$$=\mathbf{H}.\hat{\mathbf{x}}+\mathbf{n} \quad (2)$$

where INFFT is an N×N IFFT matrix that modulates the input symbols. Also O(1) and O(2) are zero matrices of size

(N+T-1)×(N+ν-T+1-L+ν) and (N+T-1)×(N+ν-K) respectively. $\bar{\mathbf{h}}=[h_L,...,h_0,...h_{-k}]$ is the channel impulse response in reverse order, yi and ni for i= 1, 2,…, N, are

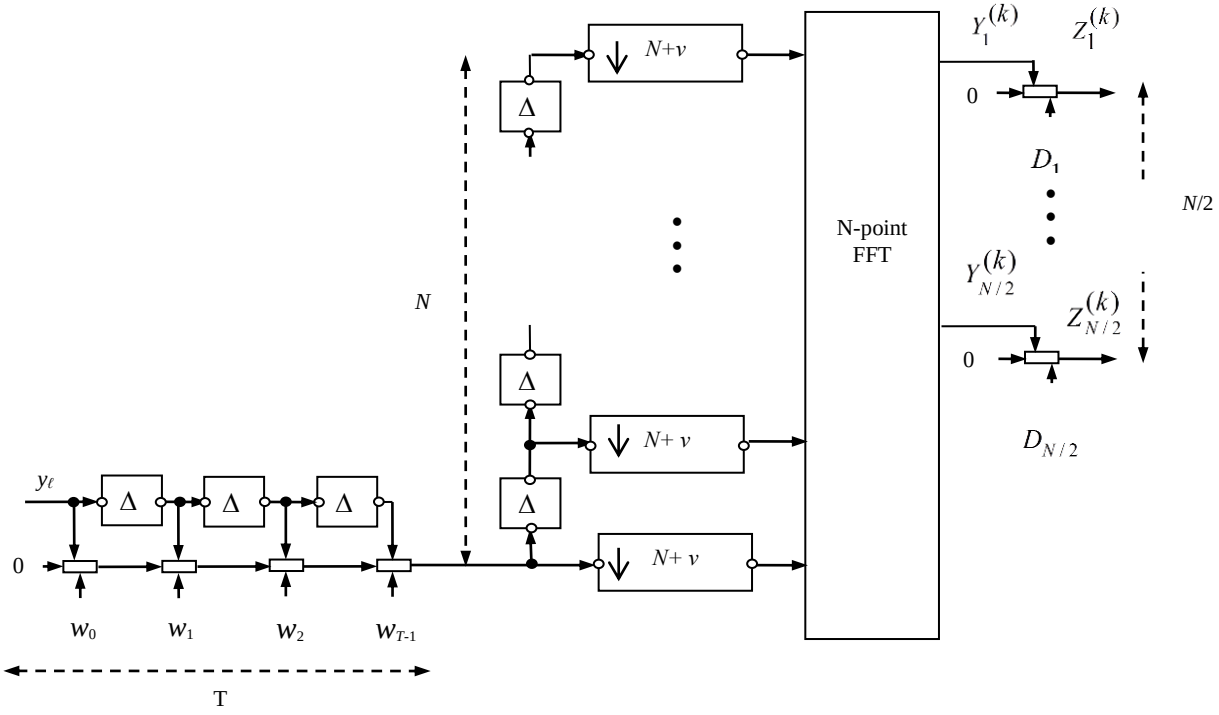


Fig. 3. T-tap TEQ equalizer for OFDM receiver

the ith component of the received and noise vectors respectively. The conventional receiver with TEQ is based on the following operation

$$\begin{bmatrix} Z^{(k)} \\ . \\ . \\ . \\ Z_N^{(k)} \end{bmatrix} = \begin{bmatrix} D_1 & \mathrm{O} & . & . & . & 0 \\ O & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & . & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & . & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & . & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & . & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & D_N \end{bmatrix} \overbrace{.F_{NFFT}.(\mathbf{Y}.\mathbf{W})}^{1FFT} \tag{3}$$

where Y matrix is a Toeplitz matrix and is defined as

$$\mathbf{Y} = \begin{bmatrix} y_{k.s+\nu+1} & y_{k.s+\nu} & \cdots & y_{k.s+\nu-T+2} \\ y_{k.s+\nu+2} & y_{k.s+\nu+1} & \cdots & y_{k.s+\nu-T+3} \\ . & & & \\ . & & & \\ . & & & \\ y_{(k+1).s} & y_{(k+1).s-1} & \cdots & y_{(k+1).s-\mathrm{T}+1} \end{bmatrix}_{N\times T} \tag{4}$$

The main idea of per tone equalization is transferring TEQ from time domain to frequency domain which is performed the following permutation for each tone i

$$Z_i^{(k)} = D_i.row_i(F_{NFFT})(\mathbf{Y}.\mathbf{W}) = row_i(\underbrace{F_{NFFT}}_{TFFT}.\mathbf{Y}).\underbrace{\mathbf{W}.D_i}_{\mathbf{W}_i} \tag{5}$$

In this work, the per-tone equalizer, which is shown in Fig. 4, is used in the OFDM receiver part and instead of the well-known algorithms such as the LMS, the MMA and S-MMA algorithms are used for updating the per-tone equalizer taps. The CP block through a serial to parallel convertor, as shown in Fig. 4, is removed. Each CP bit and the corresponding bit are compared for determining the channel effects on the data bit stream [11, 16]. In this figure Δ=N+v is the OFDM symbol length, where N is symbol size and v is the CP length. Also Vi,l is the coefficient of the per-tone equalizer and finally ↓N+v denotes down-sampling with period of N+v samples. The modified per tone equalizer is defined as $\mathbf{V}_i = [\nu_{i,o},...,\nu_{i,T-1}]^T$ and therefore Wi coefficients will be computed as

$$\begin{bmatrix} v_{i,o} \\ v_{i,1} \\ \cdot \\ \cdot \\ \cdot \\ v_{i,T-1} \end{bmatrix} = \begin{bmatrix} 1 & \alpha^{i-1} & \cdots & \cdots & \alpha^{(i-1)(T-1)} \\ 0 & 1 & \ddots & \ddots & \vdots \\ \cdot & \ddots & \ddots & \ddots & \vdots \\ \cdot & & \ddots & \ddots & \alpha^{i-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} W_{i,o} \\ W_{i,1} \\ \cdot \\ \cdot \\ \cdot \\ W_{i,T-1} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -\alpha^{i-1} & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & \vdots \\ \cdot & \ddots & \ddots & \cdot & 0 \\ \cdot & & \cdot & \cdot & -\alpha^{i-1} \\ 0 & \cdot & \cdot & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} w_{i,o} \\ w_{i,1} \\ \cdot \\ \cdot \\ \cdot \\ w_{i,T-1} \end{bmatrix} \quad (6)$$

The Eq. (6) is written as recursive form as bellow

$$v_{i,t+1}.\alpha^{i-1} + w_{i,t} = v_{i,t} \quad (7)$$

For tone i=1,…,N with t = 0,…,T-2

$$v_{i,T-1} = w_{i,T-1} \quad , \quad \alpha = e^{-j2\pi(1/N)} \quad (8)$$

$$\overline{\mathbf{w}}_i^T = [w_{i,T-1} \ldots w_{i,0}]$$

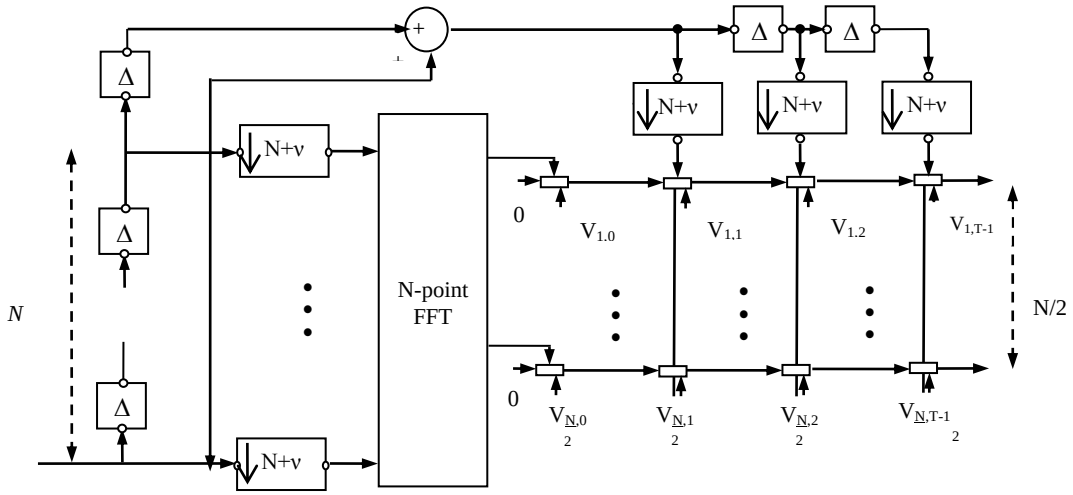


Fig. 4. Per-tone equalizer for OFDM receiver

4. Analysis of the CP Insertion in the OFDM Modulation

For considering the effects of the CP insertion, first we define the OFDM symbol in the baseband as

$$x_l[n] = \sum_{i=-N/2}^{N/2-1} a_{i,l} e^{j\frac{2\pi}{N}in} \quad s[n] \quad (9)$$

whereai,l denotes the complex symbol modulating of the ith subcarrier. The s[n] is a time rectangular window function in the interval [0,M], where M is the OFDM symbol period, and N is the number of subcarriers.

After the CP insertion with the length of Ng, the lth discrete time domain of the OFDM symbol $\tilde{x}_l[n]$ will be

$$\tilde{x}_l[n] = \sum_{i=-N/2}^{N/2-1} a_{i,l} e^{j\frac{2\pi}{N}i(n-N_g)} \quad s[n] \quad (10)$$

The channel impulse response with the channel tap weight coefficients hl and the symbol period T is

$$h(t) = \sum_{l=0}^{L} h_l \delta(t - \frac{lT}{N}) \quad (11)$$

Therefore, the output of the OFDM transmitter will be

$$x(t) = \sum_m \sum_{i=-N/2}^{N/2-1} a_{i,m} e^{j\frac{2\pi i}{T}(t-mT)} s(t-mT) \quad (12)$$

The received baseband signal at the input of the OFDM receiver will be as the linear convolution of the transmitted signal and the channel impulse response as bellow

$$u(t) = h(t) * x(t) = \sum_{l=0}^{L} h_l x(t - \frac{lT}{N})$$

$$= \sum_{l=0}^{L} h_l \sum_m \sum_{i=-N/2}^{N/2-1} a_{i,m} e^{j\frac{2\pi i}{T}(t-\frac{lT}{N}-mT)} s(t - \frac{lT}{N} - mT) \quad (13)$$

The k'th subcarrier of the m'th demodulated OFDM symbol will be

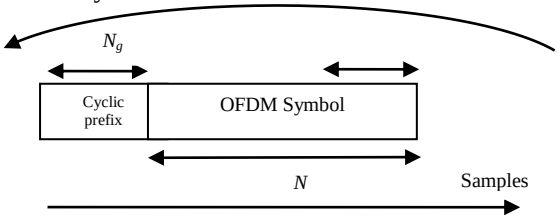


Fig. 5. Insertion of GI

$$u_{k',m'} = \frac{1}{N}\sum_{l=0}^{L} h_l \sum_{m} \sum_{i=-N/2}^{N/2-1} a_{i,m} e^{-j\frac{2\pi li}{N}} \times \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}((i-k')n+(m'-m)(i-k')N)} \times s\left(\frac{nT}{N} - \frac{lT}{N} + (m'-m)T\right) \quad (14)$$

It was proved [12] that the Eq. (14) will be rewriten as

$$u_{k',m'} = \sum_{l=0}^{L} h_l e^{-j\frac{2\pi k'l}{N}}\left(1-\frac{l}{N}\right)a_{k',m'} + \frac{1}{N}\sum_{\substack{i=-N/2 \\ i\neq k'}}^{N/2-1} a_{i,m'} \sum_{l=0}^{L} h_l e^{-j\frac{2\pi li}{N}} \sum_{n=l}^{N-1} e^{-j\frac{2\pi}{N}(i-k')n} + \frac{1}{N}\sum_{i=-N/2}^{N/2-1} a_{i,m'-1} \sum_{l=0}^{L} h_l e^{-j\frac{2\pi li}{N}} \sum_{n=0}^{l-1} e^{-j\frac{2\pi}{N}(i-k')(n+N)} \quad (15)$$

In Eq. (15), the first term is the desired data, the second term represents the ICI caused by the other subcarriers belonging to the current OFDM symbol. Finally, the third term represents the ISI caused by the subcarriers of the previous OFDM symbol [12].

The ISI can be avoided by the insertion of a guard interval (GI) at the beginning of each OFDM symbol. The GI, with Ng length, should be longer than the maximum possible of the channel impulse response length. In order to avoid ICI, the last part of the OFDM symbol can be added to the beginning of the symbol, as shown in Fig. 5. This part is called the CP. After inserting the CP, the Eq. (15) will change to

$$u_{k',m'} = \sum_{l=0}^{L} h_l e^{-j\frac{2\pi k'l}{N}} a_{k',m'} \quad (16)$$

which contains only the desired symbol, free of the ICI and ISI impairments. Therefore, by inserting a GI longer than the maximum delay spread of the channel and by cyclically extending the OFDM symbol over the GI, both the ICI and ISI eliminated completely and the channel appears to be flat fading for each subcarrier [11, 12].

5. S-MMA Equalization Performance Analysis

The relation between the transmitted symbol x(m) and the received signal u(m), as shown in Fig. 6, will be

$$u(m) = \sum_{i=0}^{L-1} h_i x(m-i) + n(m) \quad (17)$$

where hi is the ith tap of the channel impulse response with length L and n denotes the AWGN noise.

a (m) $x(m)$ $u(m)$ $y(m)$

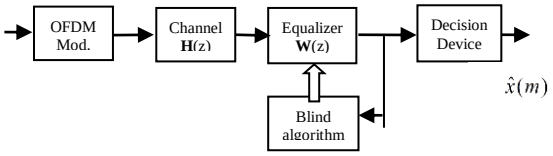


Fig.6 .Blind equalizer block diagram

Least mean square (LMS) is a well-known algorithm for updating different adaptive filter. For the LMS algorithm, the estimation error is expressed as

$$e(m) = x(m) - y(m) = x(m) - \hat{\mathbf{w}}^H(m)\mathbf{u}(m) \quad (18)$$

where $\hat{\mathbf{w}}(m)$ is the estimation of the tap-weight vector at iteration m and H denotes the Hermitian operator. In this case, the LMS cost function will be

$$J = J_{\min} + 2\,\mathrm{Re}\left\{E\left(e_o^*(m)\boldsymbol{\varepsilon}_o^H(m)\mathbf{u}(m)\right)\right\} + E\left(\boldsymbol{\varepsilon}_o^H(m)\mathbf{u}(m)\mathbf{u}^H(m)\boldsymbol{\varepsilon}_o(m)\right) \quad (19)$$

where $J_{\min}$ is the minimum mean-square error of Wiener filter and $\boldsymbol{\varepsilon}_o(m)$ is the zero-order weight-error vector of the LMS filter [20]. The LMS tap updating algorithm is

$$\mathbf{w}(m+1) = \mathbf{w}(m) + \mu\mathbf{u}(m)e^*(m) \quad (20)$$

where w(m) is the tap-weight vector at iteration m and μ is the step size and u(m) is the received signal vector or the LMS input vector [20].

The CMA is a special case of Godard's family of blind equalization algorithms [21]. Its cost function is only amplitude-dependent, and knowledge about the signal constellation is dismissed. For signal constellation which all signal points have the same magnitude, the performance of CMA is reasonable [8]. Many MMA have been presented in the past to overcome the misadjustment caused by the CMA. Some of these MMA schemes, specifically for QAM constellations, fix the phase offset error without needing any rotator at the end of the equalizer stage. The MMA minimizes the dispersion of real and imaginary parts, yR and yI, of y(n) separately [22]. The MMA, unlike the CMA, ignores the cross term yRyI between the in-phase and quadrature components. As a result, the MMA cost function is not a two-dimensional cost function and it is pseudo two-dimensional because it contains yR(n) and yI(n) only [8]. The MMA cost function and its parameters are given as

$$J = E\left\{\left(y_R^2(m) - R_R\right)^2 + \left(y_I^2(m) - R_I\right)^2\right\} \quad (21)$$

$$R_R = \frac{E[x_R^4]}{E[x_R^2]} \quad , \quad R_I = \frac{E[x_I^4]}{E[x_I^2]} \quad (22)$$

which yR(m) and yI(m) are the real and imaginary parts of y(m). The corresponding MMA tap updating algorithm is

$$\mathbf{w}(m+1)=\mathbf{w}(m)+\mu\begin{Bmatrix}(y_R(m)\left(R_R-y_R^2(m)\right)\\+\ j.y_I(m)\left(R_I-y_I^2(m)\right)\end{Bmatrix}.\mathbf{u}^*(m) \quad (23)$$

In this paper, we propose the S-MMA for using in the OFDM applications which employing QAM signals. The S-MMA cost function satisfies a number of desirable properties, including multiple-modulus, symmetry, and (almost) uniformity. The S-MMA cost function exhibits a much lower misadjustment compared to CMA and MMA [8]. The proposed S-MMA algorithm is devised by embedding the sliced symbols in the dispersion constants [8]. The S-MMA cost function is

$$J=E\begin{Bmatrix}\left(y_R^2(m)-|\hat{x}_R(m)|^c R_R\right)^2\\+\left(y_I^2(m)-|\hat{x}_I(m)|^c R_I\right)\end{Bmatrix}^2 \quad (24)$$

where $\hat{x}(m)$ is the predicted symbol and c is a positive constant. The S-MMA tap updating is

$$\mathbf{w}(m+1)=\mathbf{w}(m)$$
$$+\mu\begin{Bmatrix}(y_R(m)\left(|\hat{x}_R(m)|^c R_R-y_R^2(m)\right)\\+\ j.y_I(m)\left(|\hat{x}_I(m)|^c R_I-y_I^2(m)\right)\end{Bmatrix}.\mathbf{u}^*(m) \quad (25)$$

The S-MMA update mechanism is aware of the dispersion of y(n) away from the closest symbol $\hat{x}(m)$ in some statistical sense. The performance of an equalization algorithm maybe measured as the bit error rate (BER), the convergence rate and the residual ISI[8]. In this paper, the performance is measured by BER criteria.

6. Simulation Results

In this work, three tap updating algorithms (well-known LMS, MMA and S-MMA) are applied to the OFDM multicarrier modulation. For comparison, six standard channels, SUI[1] through SUI[6], with length of L=18 taps and AWGN (with mean=0) noise are employed. For transmission efficiency, the CP length was set to be smaller than the channel length i.d. v=16, and hence the system had ISI impairment and AWGN noise simultaneously.

For each experiment, the BER is obtained from the ensemble average of 1000 independent Monte Carlo experiments. Because the results for SUI[1] through SUI[6] channels are almost the same, we have shown the results only for SUI[1] in Fig. 7. It is clear that the S-MMA has a much lower BER than the LMS and MMA algorithms, especially for high SNR's.

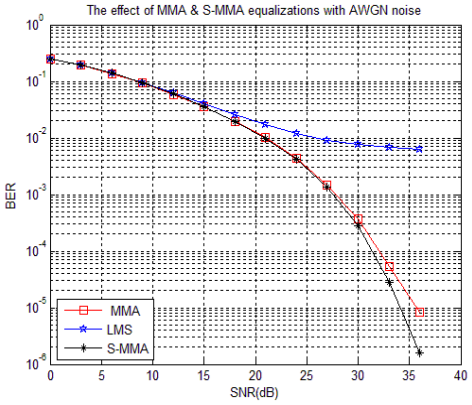


Fig.7. The effects of S-MMA on SUI[1] channel with AWGN noise

7. Conclusions

In this work, the S-MMA adaptive equalization was introduced for OFDM applications. The performance of the MMA and S-MMA was contrasted against the well-known LMS equalization in per-tone equalizer for SUI channels with the AWGN noise. For transmission efficiency, the length of the CP was set to be smaller than the channel length, and hence the system had simultaneous ISI impairment. Both analysis and simulations results show the gains and clearly verify that the S-MMA equalization, with an insufficient length of CP, has a lower BER than the most well-known LMS across all channel SNR's. Thus, the S-MMA equalization is a suitable candidate to replace for the LMS equalization in the OFDM modulation.

References

[1] C. Chong-Yong, F. Chin-Chun, C. Chii-Horng and C. Ching-Yung, "Blind Equalization and System Identification", Springer, 2006.

[2] G. Kurt, Karabulut, C. Sahin, "Channel estimation error compensation in OFDM based multi-carrier systems", Waveform Diversity & Digital Radar Conference, IET 2008, pp. 1 –4.

[3] S. Ghazi-Maghrebi, M. Lotfizad, and M. Ghanbari, "Comparison performance of different constellations with FHT in ADSL system based on DMT", IEEE, DSP2007, Cardiff, UK, 2007, pp.335-338.

[4] D.D. Falconer, and F.R. Magee, "Adaptive channel memory truncation for maximum

likelihood sequence estimation", Bell System Technology Journal, 1973, pp. 1541–1562.

[5] J.S. Chow, and J.M. Cioffi, "A cost-effective maximum likelihood receiver for multicarrier systems", Proc, IEEE international Conference on Communications, IEEE ICC 1992, Chicago, IL, USA, 1992, pp.948–952.

[6] K.W. Cheong, and J. M. Cioffi, "Precoder for DMT with insufficient cyclic prefix", IEEE international Conference on Communications. IEEE ICC 1998, Atlanta, Georgia, USA, 1998, pp. 339–343.

[7] M. Malkin, C.S. Hwang, and J.M. Cioffi, "Transmitter Precoding for Insufficient-Cyclic-Prefix Distortion in Multicarrier Systems", IEEE Vehicular Technology Conference 2008, Marina Bay, Singapore, 2008, pp.1142 – 1146.

[8] S. Abrar and R. A. Axford, "Sliced Multi-modulus Blind Equalization Algorithm", ETRI Journal, Vol. 27, No.3, Jun 2005, pp. 257-266.

[9] C.R. Johnson, P. Schniter, T.J. Endres, J.D. Behm, and D.R. Brown, "Blind equalization Using the constant Modulus Criterion", A review, Proceeding of IEEE Vol. 86, Oct. 1998, pp. 1927-1950.

[10] J.R. Treichler and B.G. Agee, "A New Approach to Multipath Correction of constant Modulus Signals," IEEE Transaction Acoust. Speech Signal Processing, vol. ASSP-31, 1983, pp. 459–471.

[11] K. V. Acker, M. Moonen, and T. Pollet, "Per-Tone equalization for DMT-based systems", IEEE Transactions on Communications, Vol. 49, No. 1, 2001, pp. 109-119.

[12] S. Ghazi-Maghrebi, H. Motahayeri, K. Avanesian, "A New Mathematical Analysis of the Cyclic Prefix Effect On Removing ISI and ICI in DMT Systems", IEEE TENCON international conference, 2011, pp. 626-630.

[13] J. Sivadasan, "Effectiveness of Orthogonal Frequency Division Multiplexing Technique for Wireless Telecommunication Systems", International Conference on Information and Communication Technology in Electrical Sciences (ICTES 2007), IET-UK, 2007, pp.758-761.

[14] H. Hu, H. H. Chen, K. Guo, and M. Weckerle, "Cross-layer adaptive resource allocation for OFDM systems with hybrid smart antennas", IET Communications, Vol. 1, No. 5, 2007, pp. 831 – 837.

[15] Y. Wei, Y. Weisheng, Y. Zhenhua, Y. Ziqiao, and H. Jing, "Implement of signal transmitter for OFDM in underwater acoustic communication based on DDWS", IET Conference on Wireless, Mobile and Sensor Networks, (CCWMSN07). Dec. 2007, pp. 1005 – 1009.

[16] S. S. C. Rezaei, M. Pakravan, per tone equalization analysis in DMT based systems, IEEE, 2004, pp. 530-542

[17] L. Sun, A. Sano, W. Sun, and A. Kajiwara, "Channel identification and interference compensation for OFDM system in long multipath environment", Signal Processin. 2009; Vol.89, 2009, pp. 1589-1601.

[18] S. Ghazi-Maghrebi, M. Lotfizad, and M. Ghanbari, "The better performance of the new non-rectangular QAM with FHT in ADSL system based on DMT without cyclic prefix", IEEE, DSP2007, Cardiff, UK, 2007, pp. 335-338.

[19] Y.J. Kou, W.S. Lu, and A. Antoniou, "A new peak-to-average power-ratio reduction algorithm for OFDM systems via constellation extension", IEEE Transactions on Wireless Communications, 2007, pp. 6.

[20] S. Haykin, Communication systems, John Wiley & Sons, Inc., 4th Edition, 2001.

[21] D.N. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communications systems," IEEE Trans. Communication, vol. COM-28, Nov. 1980, pp. 1867–1875.

[22] J. Yang, J.J. Werner, and G.A. Dumont, "The Multi-modulus Blind Equalization and its Generalized Algorithms," IEEE J. Selected Areas Communication, Vol. 20, No. 5, Jun. 2002, pp. 997–1015.

A Conflict Resolution Approach using Prioritization Strategy

Hojjat Emami*
Department of Computer Engineering, Islamic Azad University, Miandoab Branch, Iran
hojjatemami@yahoo.com
Kamyar Narimanifar
Department of Civil Engineering, Islamic Azad University, Miandoab Branch, Iran
kamyar1360@yahoo.com

Received: 27/Sep/2012 Accepted: 16/Dec/2012

Abstract
In current air traffic control system and especially in free flight method, the resolution of conflicts between different aircrafts is a critical problem. In recent years, conflict detection and resolution problem has been an active and hot research topic in the aviation industry. In this paper, we mapped the aircrafts' conflict resolution process to graph coloring problem, then we used a prioritization method to solve this problem. Valid and optimal solutions for corresponding graph are equivalent to free conflict flight plans for aircrafts in airspace. The proposed prioritization method is based on some score allocation metrics. After score allocation process, how much the score of an aircraft be higher its priority will be higher and vice versa how much the score of an aircraft be lower its priority will be lower. We implemented and tested our proposed method by different test cases and test results indicate high efficiency of this method.

Keywords: Air Traffic Control, Free Flight, Conflict Detection and Resolution, Graph Coloring Problem, Prioritization Method.

1. Introduction

Air traffic management is a very difficult, dynamic and complex problem [1]. Nowadays, the airspace management system has high flight capacity, therefore control of existing enormous volume of flights is very difficult [2, 3]. Current air transportation systems are faced with many problems. The aviation industry introduced a new approach called free flight for solving various problems in current air traffic management [4, 5]. Free flight or user preferred trajectories, is an innovative method introduced to improve the safety and efficiency of the national airspace system. Currently free flight method is technically practical because exist its required technologies. Free flight method has many advantages such as less fuel consumption, reduction of flight times, flights' delays and reduction the workload of air traffic controllers. Despite many advantages of this method, free flight imposes some problems for air traffic management system that one of the important of them is the conflict problem between different aircrafts' flights [6, 7]. Conflict detection and resolution is one of the major and fundamental challenges in safe, efficient and reliable air traffic management system. In this paper, conflict is defined as: "*the event in which two or more than two aircrafts experience a loss of minimum separation from each other*" [8]. Also the conflict detection process is defined as: "*the process of deciding when conflict between aircrafts will occur*", and conflict resolution process is considered as: "*specifying what action and how should be to resolve conflicts*" [8]. Annually Conflicts between different aircrafts causes many losses for aviation industry.

Generally many researchers have been presented various models to automate conflict detection and resolution system (e.g. in [9, 10, 11]). In reference [8] Kuchar and Yang provided a review of some of proposed conflict detection and resolution modeling methods. Also in reference [12] we presented an overview of a number of multi-agent conflict detection and resolution methods.

This paper presented a conflict resolution methodology for aircrafts' flights in airspace. This method has high efficiency, flexibility and reliability. In this paper we used concept of graph coloring problem [13]. In fact we mapped congestion area to a corresponding state space graph. Each vertex of this graph indicates an aircraft in airspace and each edge of this graph indicates a predicted conflict between two aircrafts in future times. Also in this paper we proposed a new prioritization method for solving

* Corresponding Author

conflicts problem. By using prioritization algorithm we make a priority list for aircrafts that exist in congestion area. In our proposed model, after mapping congestion area to a corresponding graph we used this priority order for coloring this graph (i.e. solving conflicts between aircrafts). A valid and optimal coloring for this graph is equal to a new free conflict flight plan. The simulation results indicate this algorithm has high efficiency and it is sound.

The organization of this paper is as follows. In Section 2, graph coloring problem is described. Section 3 describes prioritization method. In Section 4 we explain our proposed model. Section 5 discusses experiments and simulation results and finally in Section 6 we make some conclusion and present an outlook of future works.

2. Graph coloring Problem

Graph coloring problem (GCP) [13, 14] involves labeling each vertex of given graph G, so that no two adjacent vertices have the same colors. One of the goals of graph coloring problem is to minimize the number of colors used in the coloring process. Graph coloring problem is a practical method and is a NP-hard problem [15]. Graph coloring problem arises naturally in many real world application fields such as register allocation, frequency assignment, time scheduling, and circuit board testing.

Assume an undirected graph G = (V, E) with a set of vertices V, and a set of edges E, a k-coloring of G include assigning a color to each vertex of V such that no two adjacent vertices have the same color. In other word, a k-coloring of G = (V, E) can be stated as a function C from V to a set of colors K such that |K|=k and C (u) $\neq$ C (v) whenever E contain an edge (u, v) for any two vertices u and v of V. The minimal number of colors k for which a k-coloring exists is called the chromatic number of G. Optimal coloring is one that uses exactly the predefined chromatic number for that graph.

For example assume we have a graph G as illustrated in fig. 1.a. This graph has four nodes (i.e., |V| = 4) and four edges (i.e., |E| = 4). The chromatic number for this graph is equal to two (i.e., |K| = 2). For coloring this graph we use two colors (red and green). The colored graph indicated in fig. 1.b.

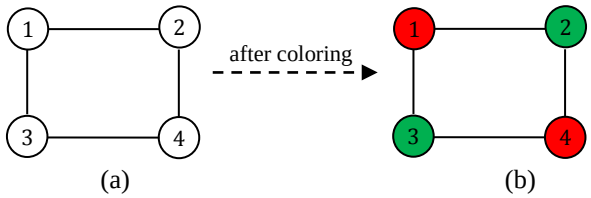


Fig. 1 A simple example of graph coloring process.
(a) graph G before coloring; (b) graph G after coloring.

There are many methods presented for Graph Coloring Problem such as: evolutionary methods (e.g. genetic algorithm [16, 17]), local search algorithms (e.g. Tabu search [18] or simulated annealing [19] and etc). In this paper to solve the graph coloring problem we used a prioritization method described in next section.

3. Prioritization Method

In this section, we introduce a prioritization method to solve conflicts between different aircrafts. We assign a (unique) priority for each aircraft based on its scores. The scores of each aircraft are specified based on situation of that aircraft in airspace. So that in priority allocation process if an aircraft has higher score, its priority will be higher and vice versa if total score of an aircraft be lower its priority will be lower.

We used simple score allocation criterions for each aircraft. These criterions are as follows:

- The score of an aircraft will increase if it had least distance to destination.
- This criterion is defined for prevention of congestion in airspace.
- The score of an aircraft will increase if it flies in the satisfactory weather condition.
- This criterion defined to consider environment conditions.
- The score of an aircraft will increase if it had higher speed (under valid speed).
- This criterion causes the traffic rate increases.
- The score of an aircraft increases, when the aircraft flies at higher altitude (under valid altitude).
- When aircrafts fly on higher altitude their fuel consumption decreases.

- The score of an aircraft increases, when its distance (horizontal or vertical) from the other aircrafts is higher.

In conflict resolution process, the aircraft with a lower priority must change its original flight path in order to prevent of occurring conflicts. In fact, we use a hierarchy method to resolve conflicts. Perhaps, this prioritization method seems very similar to the greedy method but this method is general and reasonable. For example, when an aircraft is closer to its destination, and had appropriate speed and minimum deviation from the mainstream, it must be serviced in first and then other aircrafts must be serviced. Although, in this case starvation state occurring is not unexpected but we can avoid this problem by allocating scores to the aircrafts that for long time are on the flight paths, so these aircrafts also service immediately in least possible time. It is worthwhile to mention that we can use the prioritization method to solve conflicts without using of graph coloring problem.

4. Our proposed model

The block diagram of our proposed model is shown in fig. 2. As shown in fig. 2, firstly the traffic environment must be monitored and appropriate traffic information must be collected. This information provides an estimation of current traffic situation (such as, the position, direction, destination and speed of the aircraft).

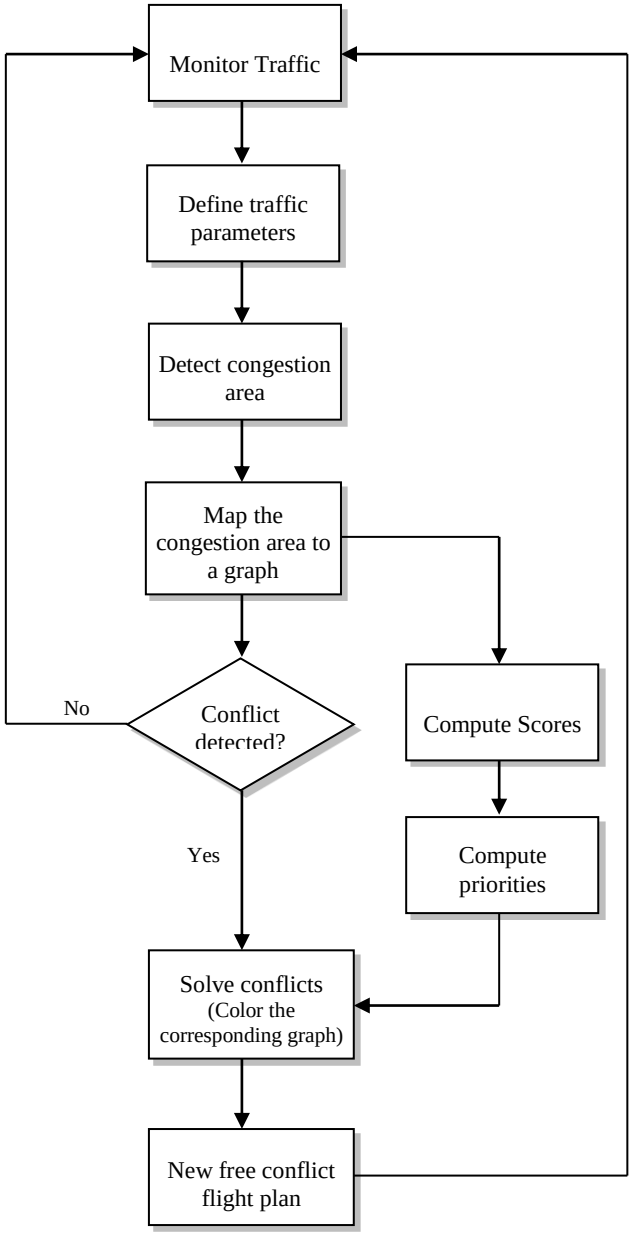


Fig. 2 our proposed model.

After this stage we specify domain of congestion area and then we map this area to a corresponding graph. Each vertex of this graph indicates an aircraft in congestion area and each edge in this graph indicate a predicted conflict in future states. In other words in this stage we map the congestion area to a state space graph.

In next step, the scores of aircrafts in congestion area are computed based on some score allocation criterions. Then based on allocated scores to aircrafts, the priority of each aircraft is specified.

In the third stage, the corresponding graph is colored using prioritization method. The output of the algorithm is an optimal and reliable coloring (an efficient solution for solving conflicts between aircrafts in congestion area).

If there is no collision, the algorithm ends. Then, we send the new free conflict flight plan to the aircrafts on flight paths. Here we emphasize that our proposed model can interact with innovative technologies (such as multi-agent systems technology) to conflicts detection and resolution in air traffic management and also in ground traffic and related applications.

5. Experiments and Results

To evaluate our proposed model explained in previous sections, we used randomly generated test cases. Each test case consists of several aircrafts with different or same velocity, altitude, position and heading. These scenarios based on a test case used by krozel et al. [20, 21], and hill et al. [22], comprise of two concentric circles in open airspace. All aircrafts appears at random points on the outer circle with 100 miles, and destination of each of aircrafts at random point on inner circle with 80 miles.

We have used supposed test cases to test our proposed conflict resolution model, but we attempted to test samples very similar to the real world patterns. These test cases provide a wide range of conflict patterns that any conflict detection and resolution method must be evaluated across these test cases. Conflict resolution maneuvers used in our proposed model include small altitude and heading changes.

Table 1 shows the average of system efficiency from five simulation runs of the proposed model at each reported density. In table 1, column 1 indicates the number of aircrafts in airspace, column 2 indicates the average number of predicted conflicts and last columns indicate the efficiency of our proposed conflict detection and resolution model. The results of simulations show proposed model has high efficiency; this means our proposed model decrease flight delays and increases passengers' comfort.

Here we used a simple efficiency metric. This metric is same as the metric used in reference [20, 21]. This metric measure the degree to which an aircraft are able to track direct and optimal flight path from origin to its destination. In fact in conflict resolution process some aircrafts (in general aircrafts with lower priorities) should be deviate from their optimal and ideal mainstream in order to prevent of conflicts. In conflict resolution process our proposed model tries to decrease the number of deviations for aircrafts.

For a test case with N aircraft at the end of simulation run the efficiency of the proposed conflict detection and resolution model is as Eq.(1). In the ideal system the efficiency value equals to 1. As traffic density and number of conflicts increases the value of efficiency metric decreases.

$$efficiency = \frac{1}{N}\left(\sum_{i=1}^{N} \frac{t_{ideal}}{t_{ideal} + t_{delay}}\right) \quad (1)$$

$$t_{delay} = t_{actual} - t_{ideal} \quad (2)$$

t_{ideal} = the ideal flight time for aircraft "i" (specified when the aircraft first arrived in simulation)

t_{delay} = the delay time for aircraft "i"

t_{actual} = the actual flight time for aircraft "i"

Table 1: Result for the random flight scenarios after five simulation runs.

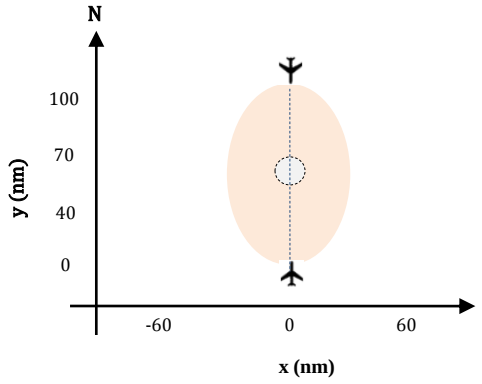
Aircrafts	Predicted conflicts	Efficiency (%)
24	18	92.6
20	10	95
16	8	95.8
12	7	96.1
10	6	97
8	5	98
6	4	98.8
4	2	99.5
2	1	99.8
2	0	1

In fact our proposed model is a preliminary and abstract conflict resolution methodology;

nonetheless this model has high efficiency and works as better.

5.1 Example Scenario

To illustrate the process of proposed prioritization method, consider two-aircraft scenario depicted in fig. 3. This example involves two aircrafts A1 and A2 that these aircrafts are headed directly their destination. We supposed these aircrafts restricted to fly in same altitude. As shown in fig. 3, if aircraft A1, A2 continue on their current heading without any deviation from their mainstreams, the aircrafts will collide. In fig. 3, if aircraft A1 and A2 continue on their previous trajectories after 7.5 minutes will collide. Aircraft A1 and A2 have 500 mph speed. These two aircrafts fly at the same altitude. Aircraft A1 has 140nm distance to its destination and distance to destination of aircraft A2 is 200nm.



A1:	A2:
Airspeed: 500 mph	Airspeed: 500 mph
Distance to Destination: 140 nm	Distance to Destination: 200 nm

Fig. 3 Example Scenario

In our proposed model we used nominal state projection method to predict and detect possible conflicts that going to occur. In first step to resolve conflicts we compute scores of aircrafts and then allocate a (unique) priority for each aircraft. For instance, here we only used "distance to destination" score allocation metric. According to this metric the score of aircraft A1 and A2 respectively is equal to 2.43 and 1.7. As we mentioned in our proposed model the aircrafts which had higher score will have higher priority and the aircrafts that had lower score will have lower priority. So aircraft A1 has higher score and subsequently its priority is equal to 1, and aircraft A2 has priority order 2. The lesser number indicates the high priority. Then to resolve predicted conflict between two aircrafts we send a command to aircraft with lower priority to deviate from its original trajectory in order to prevent collision. The aircraft which has lesser priority after receiving the deviation command, according to its conditions reply to other aircrafts acceptance or rejection message. In this scenario aircraft A2 has lower priority therefore this aircraft deviates from its mainstream, hence the predicted conflict resolved.

6. Conclusions

In this paper we proposed a systematic conflict resolution approach using graph coloring problem concept and prioritization method. Also in this paper we introduced some score allocation criterions and allocated a priority for each aircraft based on these criterions. The proposed prioritization method is natural, sound and flexible. This method considers traffic conditions to make the best decisions in critical environmental conditions for solving conflicts between aircrafts.

Simulation results on different test cases indicate the prioritization method can offer good efficiency and safety for resolving conflicts in free flight air traffic control method. Air traffic control is a dynamic problem, so that one problem in proposed prioritization method is that we can't accurately adjust the weight of different score allocation metrics, therefore in priority assigning process may be allocated priorities not correct.

Future work will comprise the extension of prioritization method to have high adaptability with traffic situations. Also we will focus on using multi agent systems with prioritization method to present a comprehensive model with high efficiency for conflict detection and resolution in air traffic management system.

References

[1] K. Tumer and A. Agogino, "Improving air traffic management with a learning multiagent system", IEEE Intell. Syst., vol. 24, no. 1, pp. 18–21, Jan/Feb. 2009.

[2] Federal aviation regulations and aeronautical information manual, 2010 edition. 2009 asa, inc. Newcastle, Washington.

[3] Department of Transport, U.K., "Air traffic forecasts for the United Kingdom 1997," U.K. Government, Department of Transport, Tech. Rep., 1997. [Online]. Available http://www.aviation.dft.gov.uk/aed/air/aircont.htm.

[4] Radio Technical Commission for Aeronautics. Final report of RTCA Task Force 3: Free flight implementation, RTCA, Washington DC, Tech. Rep.,Oct. 1995.

[5] Federal Aviation Administration, "Free Flight Introduction", http://www.faa.gov/freeflight/ffov.htm, September 2011.

[6] Federal aviation administration, "advancing free flight through human factors", www.hf.faa.gov/docs/508/docs/freeflt.pdf, accessed 1 august 2011, 1995.

[7] J. Rong, J. Valasek, S. Geng, and Ioerger RT, "Air traffic conflict negotiation and resolution using an onboard multi agent system", Proceedings of the 21st Digital Avionics Systems Conference, 2002.

[8] J. Kuchar and C. Yang, "A Review of Conflict Detection and Resolution Modeling Methods"; IEEE Transactions on Intelligent Transportation Systems, Vol. 1, No. 4, December 2000.

[9] M. Nguyen-Duc, J. Briot, and A. Drogoul, "An application of Multi-Agent Coordination Techniques in Air Traffic Management", Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT'03), 2003.

[10] S. Wollkind, J. Valasek, and RT. Ioerger, "Automated conflict resolution for air traffic management using cooperative multi-agent negotiation", AIAA Guidance, Navigation and Control Conference, 2004.

[11] N. Archambault, and N. Durand , "Scheduling Heuristics For on-Board Sequential Air Conflict Solving", IEEE, 2004.

[12] H. Emami, F. Derakhshan, "An Overview on Conflict Detection and Resolution Methods in Air Traffic Management using Multi Agent Systems", Proceeding of AISP 2012, Shiraz, Iran, May 2012.

[13] T.R. Jensen, B. Toft, "Graph Coloring Problems", Wiley interscience Series in Discrete Mathematics and Optimization, 1995.

[14] D. Werra, "Heuristics for Graph Coloring", Computing Suppl. 7, pp. 191-208, 1990.

[15] M.R. GAREY, and D.S. JOHNSON, "Computers and intractability: a guide to the theory of NP-completeness", W.H. Freeman and Company, New York, 1979.

[16] D. E Goldberg. "Genetic Algorithms in Search, Optimization and Machine Learning". Addison-Wesley, Reading, MA, 1980.

[17] C. FLEURENT, and J.A. FERLAND, "Genetic and hybrid algorithms for graph coloring". Dans G. Laporte, I.H. Osman, (Eds.), Metaheuristics in Combinatorial Optimization, Annals of Operations Research, 63 : 437-441, 1996.

[18] M. Kubale, "Introduction to Computational Complexity and Algorithmic Graph Coloring", Gdanskie Towarzystwo Naukowe, 1998.

[19] M. CHAMS, A. HERTZ, D. de WERRA, "Some experiments with simulated annealing for coloring graphs". EJOR 32: 260-266, 1987.

[20] J. Krozel, M.Peters, K. D. Bilimoria,, C. Lee, , and J. S. B. Mitchell, "System Performance Characteristics of Centralized and Decentralized Air Traffic Separation Strategies," Fourth USA/Europe Air Traffic Management Research and Development Seminar, 2001.

[21] H. Emami, F. Derakhshan, S. Pashazadeh, "A New Prioritization Method for Conflict Detection and Resolution in Air Traffic Management". Journal of Emerging Trends in Computing and Information Sciences, VOL. 3, NO. 7, pp. 1042-1049, 2012.

[22] K. Archibald, c. Hill, a. Jepsen, c. Stirling, and l. Frost, "A satisficing approach to aircraft conflict resolution" , ieee transactions on systems, man, and cybernetics—part c: applications and reviews, vol. 38, no. 4, July 2008.

A Basic Proof Method for the Verification, Validation and Evaluation of Expert Systems

Armin Ghasem Azar*
Department of Computer and Information Sciences, Institute for Advanced Studies in Basic Sciences (IASBS), Tehran, Iran
a.ghasemazar@iasbs.ac.ir
Zohreh Mohammad Alizadeh
Department of Computer and Information Sciences, Institute for Advanced Studies in Basic Sciences (IASBS), Tehran, Iran
z.alizadeh@iasbs.ac.ir

Received: 06/Oct/2012 Accepted: 23/Feb/2013

Abstract
In the present paper, a basic proof method is provided for representing the verification, Validation and evaluation of expert systems. The result provides an overview of the basic method for formal proof such as: partition larger systems into small systems prove correctness on small systems by non-recursive means, prove that the correctness of all subsystems implies the correctness of the entire system.

Keywords: Expert System, Partition, Non-Recursive.

1. Introduction

An expert system is correct when it is complete, consistent, and satisfies the requirements that express expert knowledge about how the system should behave.

For real-world knowledge bases containing hundreds of rules, however, these aspects of correctness are hard to establish. There may be millions of distinct computational paths through an expert system, and each must be dealt with through testing or formal proof to establish correctness.

To reduce the size of the tests and proofs, one useful approach for some knowledge bases is to partition them into two or more interrelated knowledge bases. In this way the VV&E problem can be minimized [1].

2. Overview the Proofs Using Partitions

The basic method of proving each of these aspects of correctness is basically the same. If the system is small, a technique designed for proving correctness of small systems should be used. If the system is large, a technique for partitioning the expert system must be applied and the required conditions for applying the partition to the system as a whole should be proven. In addition the correctness of any subsystem required by the partition must be ensured. Once this has been accomplished this basic proof method should be applied recursively to the sub-expert systems. Once the top level structure of the Knowledge base has been validated, to show the correctness of the expert system, the following criteria must be accomplished [6]:

- Show that the Knowledge base and inference engine implement the top level structure;
- Prove any required relationships among sub-expert systems or parts of the top level Knowledge representation;
- Prove any required properties of the sub-Knowledge bases.

2.1 A Simple Example

To illustrate the basic proof method, *Knowledge Base* 1 will be proved correct in Table 1 and although this Knowledge base is small enough to verify by inspection.

2.1.1 Illustrations of Knowledge Base 1

The *Knowledge Base* 1 (KB1) has six rules. There are seven variables which can take two possible values. It is, therefore a seven dimensional, binary problem [5]. Let's focus on Rule 3 to understand the illustrations of KB1.

It has two hypotheses, and one conclusion. The hypotheses are “Do you buy lottery tickets?”=”yes”, and “Do you currently own stock”=”yes”. They are associated with the logical operator “or”. The consequent is Risk Tolerance”=”high”. This is illustrated in Figure 1. For the two variables of the hypotheses in Rule 3, there are two possible values: “yes” or “no”. The number of possible combinations of values for the variables is four. These four combinations appear in Figure 1 as four square regions defined

* Corresponding Author

by the closed boundary (defining the domain or the variables) and the line boundaries separating the possible values for each variable. Each square is a Hoffman region.

Rule 1	If "Risk tolerance" = high AND "Discretionary income exists"= yes then investment = stocks.
Rule 2	If "Risk tolerance" = low OR "Discretionary income exists" = no then investment = "bank account".
Rule 3	If "Do you buy lottery tickets" = yes OR "Do you currently own stocks" = yes then "Risk tolerance" = high.
Rule 4	If "Do you buy lottery tickets" = no AND "Do you currently own stocks" = no then "Risk tolerance"= low.
Rule 5	If "Do you own a boa" = yes OR "Do you own a luxury car" = yes then "Discretionary income exists" = yes.
Rule 6	If "Do you own a boat" = no AND "Do you own a luxury car" = no then "Discretionary income exists" = no.

Table 1: Knowledge Base 1 [7]

If variable "Do you buy lottery tickets" is assigned a value "yes", then two of the four regions are relevant. In Figure 1.a, they are shown with a hatch. The two regions corresponding to hypotheses "Do you currently own stock?"="yes" are hatched in Figure 1.b.

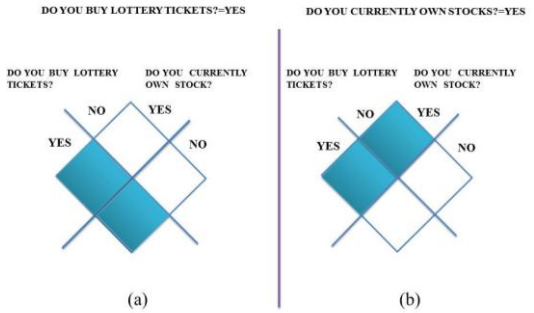


Fig. 1: Knowledge Base 1 [7]

In two dimensions, a Hoffman region is a surface as shown in this example. In three dimensions, it would be a volume.

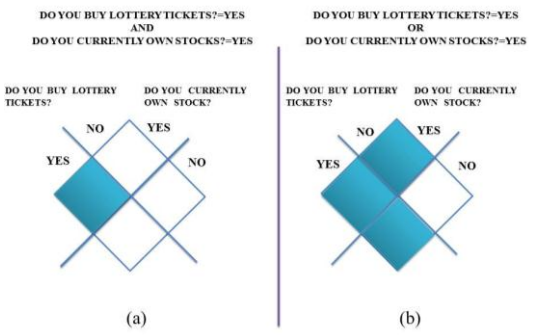


Fig. 2: Knowledge Base 1 [7]

The logical operators are "and", "or" and "not". In Figure 1.a and 1.b, the Hoffman regions corresponding to hypothesis of Rule 3 are hatched. When combined with an "and" logical operator, intersection of the two sets of Hoffman regions. This is shown in Figure 2.a.

The intersection in this case is a unique Hoffman region. In Rule 3, an "or" operator connects the two hypotheses.

In this case, the union two sets of Hoffman regions are taken, as shown in Figure 2.b.

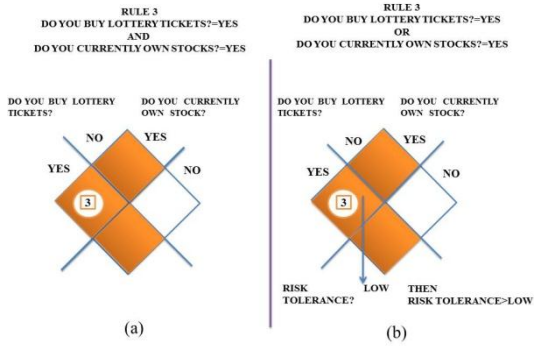


Fig. 3: Knowledge Base 1 [7]

Next, the region by the logical expression of the hypotheses is labeled with its rule. For Rule 3, the three Hoffman regions are labeled with a circled 3 as shown in Figure 3.a. Consequence for the Rule is linked to the label of the region of the hypotheses. In Figure 3.b, an arrow starts at the circled 3 and ends at the value "low" of the variable "Risk".

2.2 Step 1-Determine Knowledge Base Structure

To prove the correctness of Knowledge Base 1 (KB1), the expert Knowledge can determine that the system represents a 2-step process [3]:

- Find the values of some important intermediate variables, such as risk tolerance and discretionary income;
- Use these values to assign a type of investment.

KB1 was built using this Knowledge; therefore, it can be partitioned into the following pieces:

- A subsystem to find risk tolerance (Part of Step 1);
- A subsystem to find discretionary income (Part of Step 1);
- A subsystem to find type of investment given this Information (Part of Step 2).

2.3 Step 2-Find Knowledge Base Partition

To find each of the three subsystems of KB1, an iterative procedure can be followed:

- Start with the variables that are goals for the subsystem, e.g., risk tolerance for the risk tolerance subsystem;
- Include all the rules that set subsystem variables in their conclusions. For the risk tolerance subsystem, Rules 3 and 4 are included;
- Include all variables that appeared in rules already in the subsystem and are not goals of another subsystem;
- For the risk tolerance subsystem, include "Do you buy lottery tickets" and "Do you currently own stocks";
- Quit if all rules setting subsystem variables are in the subsystem, or else go to Step 2. For the risk tolerance subsystem, there are no more rules to be added.

Figure 4 below shows the partitioning of KB1 using this method.

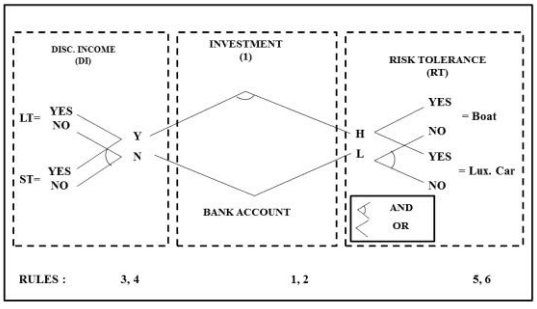


Fig. 4: Knowledge Base 1 [3]

2.4 Step 3-Completeness of expert systems

2.4.1 Completeness Step 1-Completeness of Subsystems

The first step in proving the completeness of the entire expert system is to prove the completeness of each subsystem. To this end it must be shown that for all possible inputs there is an output, i.e., the goal variables of the subsystem are set. This can be done by showing that the OR of the hypotheses of the rules that assign to a goal variable is true [7].

2.4.2 Completeness Step 2-Completeness of the entire system

The results of subsystem completeness are used to establish the completeness of the entire system. The basic argument is to use results on subsystems to prove that successively larger subsystems are complete. At each stage of the proof there are some subsystems known to be complete; initially the subsystem that concludes overall goals of the expert system will be complete. At each stage of the proof, a subsystem that concludes some of the input variables of the currently-proved-complete subsystem is added to the currently complete subsystem. After a number of steps equal to the number of subsystems, the entire system can be shown to be complete.

2.5 Step 4-Consistency of the entire system

The first step in proving the consistency of the entire expert system is to prove the consistency of each sub- system. To do this, the user must show that for all possible inputs, the outputs are consistent, i.e., that the AND of the conclusions can be satisfied.

For example, if an expert system concludes:

"temperature >0" and "temperature <100"

The AND of these conclusions can be satisfied. However, if the system concludes:

"temperature <0" and "temperature>100"

The AND of these two conclusions has to be false. It is clear that based on the input that produced these two conclusions, it is not possible for all of the system's conclusions to be true at the same time and thus the system producing these conclusions is inconsistent.

2.5.1 Consistency Step 1-Find the mutually inconsistent conclusions

The first step in proving consistency is to identify those sets of mutually inconsistent conclusions for each of the subsystems identified in the "Find partitions" step above. Some sets of

conclusions are mathematically inconsistent [2]. For example, if a system describes temperature, the set: “temperature <0”, “temperature >100” is mathematically inconsistent.

Because some sets of conclusions are inconsistent because of domain expertise, finding all sets of inconsistent conclusions generally requires expert Knowledge.

Note that if there are no mutually inconsistent conclusions in the expert system as a whole, then consistency is true by default, and no further consistency proof is necessary.

2.5.2 Consistency Step 2-Prove consistency of subsystems

If there are inconsistent conclusions in the Knowledge base as a whole, then the next step in proving consistency is to prove the subsystems consistent. This can be done by showing that no set of inputs to a subsystem can result in any of the sets of inconsistent conclusions.

2.5.3 Consistency Step 3-Consistency of entire system

The results of subsystem consistency are used to establish the consistency of the entire system. The basic argument is to use results on subsystems to prove that successively larger subsystems are consistent. At each stage of the proof, there are some subsystem known to be consistent; initially, this is the subsystem that concludes goals of the expert system as a whole. At each stage of the proof, a subsystem that concludes some of the input variables of the currently-proved-consistent subsystem is added to the currently consistent subsystem. After a number of steps equal to the number of subsystems, the entire system can be shown to be consistent [2].

2.6 Step 5-Specification satisfaction

In order to prove that KB1 satisfies its specifications, the user must actually know what its specifications are. This is a special case of the general truth that in order to verify and validate, the user must know what a system is supposed to do. Specifications should be defined in the planning stage of an expert system project [4].

To illustrate the proof of specifications it will be assumed that KB1 is supposed to satisfy:

A financial advisor should only recommend investments that an investor can afford.

As with many other aspects of verification and validation, expert Knowledge must be brought to bear on the proof process. For KB1, an expert might say that anyone can afford a savings account. Therefore, the user only has to look at the conditions under which stocks are recommended. However, that same expert would probably say that just having discretionary income does not mean that the user can afford stocks; that judgment should be made on more than one variable. Therefore, it would be reasonable to conclude that KB1 does not satisfy the above specification.

3. Conclusion

This paper has argued that V&V techniques are an essential part of the Knowledge engineering process, because they offer the only way to judge the success (or otherwise) of a KBS development project. This is equally true in the context of Knowledge management, where V&V techniques tell us whether or not the KBS can be relied upon to accurately embody the Knowledge of the human experts that supplied it.

However, examination of known studies on the effectiveness of existing KBS VV&E techniques has shown that the state of Knowledge in this area is sparse. The way to improve this situation would be by systematically gathering data from a representative set of KBS projects and V&V techniques. Without such a study, Knowledge engineering will remain very much an art and, by extension, so will the use of KBS technology in Knowledge management.

It is difficult to generalize our results to all Knowledge based systems and, of course, further evaluations of other applications are necessary to confirm (or challenge) our conclusions. However, since the method we have used minimizes the need for experts' interpretation of the faults, we can reasonably conclude that if we use an application of similar size and complexity to GIBUS, we would expect to obtain similar results. Consequently, since our application has a size and a complexity which is representative of actual practice, we would expect that consistency and completeness checking, in addition to testing, would be an effective combination of methods to validate many of the Knowledge based systems actually under development.

References

[1] Ayel M and Laurent J-P, two different ways of verifying Knowledge-based systems, Validation, Verification and Test Of Knowledge-Based Systems, Wiley, New York, Year. 1991, pp. 63-76.

[2] Bendou A, A constraint-based test data generator, EUROVAV-95, Saint Badolph, France, Year. 1995, pp. 19-29.

[3] Ginsberg A, Knowledge-based reduction: A new approach to checking Knowledge bases for inconsistency & redundancy, AAAI Vol. 88, No. 2, Year. 1988, pp. 585-589.

[4] Kirani S, Zualkernan I.A, and Tsai W.T., Comparative Evaluation of Expert System Testing Methods, Computer Science Department, University of Minnesota, Minneapolis Vol. 2, Year. 1992, pp. 92-30.

[5] Laurent J-P, Proposals for a valid terminology in KBS validation, ECAI-92, Wiley, New York, Vol. 2, Year. 1992, pp. 829-834.

[6] Lounis R and Ayel M, Completeness of KBS, EUROVAV-95, Saint Badolph, France, Vol. 2, Year. 1995, pp. 31-46.

[7] O'Leary D, Design, development and validation of expert systems: A survey of developers, Vol. 2, Year. 1991.

Prediction of Deadlocks in Concurrent Programs Using Neural Network

Elmira Hasanzad
Department of Computer Engineering, University of Kashan, Iran
elm.hasanzade@grad.kashanu.ac.ir
Seyed Morteza Babamir*
Department of Computer Engineering, University of Kashan, Iran
babamir@kashanu.ac.ir

Received: 26/Nov/2012 Accepted: 20/Feb/2013

Abstract
The dependability of concurrent programs is usually limited by concurrency errors like deadlocks and data races in allocation of resources. Deadlocks are difficult to find during the program testing because they happen under very specific thread or process scheduling and environmental conditions. In this study, we extended our previous approach for online potential deadlock detection in resources allocated by multithread programs. Our approach is based on reasoning about deadlock possibility using the prediction of future behavior of threads. Due to the nondeterministic nature, future behavior of multithread programs, in most of cases, cannot be easily specified. Before the prediction, the behavior of threads should be translated into a predictable format. Time series is our choice to this conversion because many Statistical and Artificial Intelligence techniques can be developed to predict the future members of the time series. Among all the prediction techniques, artificial neural networks showed applicable performance and flexibility in predicting complex behavioral patterns which are the most usual cases in real world applications. Our model focuses on the multithread programs which use locks to allocate resources. The proposed model was used to deadlock prediction in resources allocated by multithread Java programs and the results were evaluated.

Keywords: Detecting Potential Deadlocks, Time Series Prediction, Multithread Programs, Behavior Extraction.

1. Introduction

Multithread programs are becoming increasingly common. Since multi-core processor generation has brought more cores, developers must parallelize programs if they want to speed the program execution up. However, applying concurrency method causes some integrity and mutual exclusion issues in allocating resources. To resolve them, locking mechanism was developed. However, this mechanism leads to some other known problems like *starvation* and *deadlock* in resources allocated by concurrent systems. Detection of such errors in the program testing phase may be difficult since they often occur in the special sequence of events [1]. This is why that, these errors are sensitive to timings, workloads, compiler options and memory models. In addition, if a deadlock or *data race* in resource allocation emerges in the testing phase, it is difficult to find out its root cause; because in a multithread program, even if there is a deadlock between some threads in allocating resources at runtime, other threads still can run. The effects of such a situation can manifest itself millions of cycles after occurring the error. Deadlock is a common form of bug in software nowadays. Sun's bug database showed that 6,500 bug reports out of 198,000 contain "deadlock" [2]. Main reasons of deadlock are: (1) software systems are often written by diverse programmers; therefore, it is difficult to follow a lock order discipline in allocating resources, (2) programmers often introduce deadlocks when they fix race conditions by adding new locks and (3) using third-party software such as plug-in because the third-party software may not follow the locking discipline followed by the parent software [3]. This is why that "deadlock avoidance" techniques became unusable. Such techniques are simple in theory but so restrictive in real application.

Therefore "detecting potential deadlocks" became an acceptable method to solve deadlock problem in resources allocation. "Potential deadlock detection" techniques are Online or Offline, which Online ones try to find the concurrency errors at runtime. Such approaches mostly use a monitor to observe the program

* Corresponding Author

execution and based on the observations, they decide about the error possibility. In comparison with offline techniques, online ones have the following advantages:

1. They only visit feasible paths of program executions and have accurate views of the values [1],
2. Because of their accurate view, they generate fewer false alarms. False alarm means a fake report of an error (in our case, a deadlock),
3. They don't need considerable programmer effort,
4. These approaches are language independent meaning that the solution is not depended on features of a specific programming language.

In this paper, we demonstrate and extend a novel online potential deadlock detection approach, whose base was presented in [4]. It was based on the prediction of processes or threads behavior at runtime and dealt with reasoning about the deadlock possibility in the future. In this work, we introduce *time series* analysis approaches in configuring prediction parameters. Also, we include the environmental conditions in predicting the threads behavior to improve the correctness of obtained results. We obtained considerable improvement in detecting potential deadlocks in comparison with our previous work.

This paper is organized as follows: Section 2 overviews the related works and our proposed model is discussed in Section 3. We analyze our approach and evaluate its results in Section 4. We draw conclusions in Section 5.

2. Related works

As mentioned in the previous section, our approach is based on finding potential deadlocks in allocating resources at runtime using program behavior extraction and time series prediction. Therefore in this section, we first overview online approaches detecting potential deadlocks in resources allocated by concurrent programs. Afterwards, we discuss different approaches used time series for the prediction.

2.1 Online potential deadlock detection

Informally, in multi-threaded systems used shared memory, deadlocks in allocating resources happen when a set of threads are blocked forever; this is because each thread in the set is waiting to acquire a lock held by some thread [2]. Generally in a concurrent system, the order of acquiring and releasing locks in allocating and freeing resources can be described as a directed graph where nodes indicate locking resources so that an edge from node A to node B means the system has locked resource A and is waiting for resource B. There will be a deadlock in allocation of resources if a circle is found in the graph. Lock graphs and their variations have been used for detection of deadlocks in resources allocated by concurrent programs.

GoodLock algorithm [5] is an approach to detect potential deadlocks in multithread programs. It only detects potential deadlocks caused through interleaving locks by just two threads. To overcome this limitation, some generalized versions of GoodLock algorithm was presented in [6] and [7] which detect potential deadlocks caused by any number of threads. Their approach address programs that use bloc and non-block structured locking.

In [8], authors constructed an online lock graph and found specific paths, which named "not guarded SCC (strongly connected components)". "Not guarded SCC" indicates one or more potential deadlocks because there can be several cycles in the SCC. They tried to exhibit the deadlocks using injection of noises in the SCCs. A noise is inserted to create a delay to acquire a lock; accordingly, they raised the probability of manifesting the real deadlocks. Although this approach is based on GoodLock algorithm, its advantage over one that presented in [6] and [7] is regarding different runs. The Goodlock looks at the scope of one process run. This means, when a cycle in the graph is caused by sequences of two different runs, Goodlock can't detect.

GoodLock algorithm also was used in combination with other techniques to find the potential deadlock at runtime such as DEADLOCKFUZZER [2]. This approach consists of two phases. In the first phase, a simple variant of the Goodlock algorithm, called informative Goodlock, was used to discover cycles of potential deadlock. In the second phase, DEADLOCKFUZZER executed the program at a random schedule in order to create a real deadlock corresponding to a cycle which reported in the previous phase. In [3] "deadlock immunity" concept was introduced for avoiding occurrence of deadlocks occurred in the past. When a deadlock occurs for the first time, the deadlock information is saved in a "context" in order to avoid the similar contexts in future runs. This approach achieved "immunity" against the corresponding deadlocks. To avoid deadlock whose context has been already seen, the approach changed the schedule of threads. As the several deadlocks occur, the numbers of contexts

increases; therefore it can avoid a wider range of deadlocks. However, if a deadlock does not have a pattern similar to one that already encountered, the approach cannot avoid its occurrence.

As we mentioned in Section Introduction, all the online deadlock detection approaches share some common advantages like: language independency, accurate views of the values, fewer false alarms and programmer efforts. But online techniques suffer from some disadvantages too. The most common problem is imposing the heavy overhead at runtime, both in time or space. All the mentioned techniques try to extract some relevant traces before their real execution based on observed current execution. In fact, they *pre-run* these extracted traces to find out whether there is any deadlock in the trace or not. This phase is time consuming because of extracting and running relevant traces. Also, one of the important steps for online techniques is the code *instrumentation*. Code instrumentation means modifying the target code for runtime monitoring code behavior. This step could be time consuming too and for the legacy codes it is more difficult. Sometimes, online potential deadlock detection techniques may show deadlocks late. This leads to finding a potential deadlock when the rollback mechanism is impossible because of some preclude actions like I/O.

2.2 Time series prediction approaches

Because of weaknesses of the online potential deadlock detection techniques mentioned in previous section, we proposed a novel online approach which targets the increase of performance, decrease of instrumentation and the enhancement of the prediction [4]. In this approach, we needed to predict future members of the generated *time series* at runtime. In general, time series prediction techniques can be classified in two categories: statistical and neural network based techniques. The statistical prediction techniques such as Autoregressive (AR), Moving Average (MA) and combined AR and MA (ARIMA) [9] have several limitations, such as inefficiency for real world problems which are often complex and *nonlinear*. This is due to the fact that these techniques assume that a time series is generated by a *linear* process. Thus, they are called linear statistical predictors.

The *nonlinear* statistical predictors such as predictors, "threshold", "exponential", "polynomial" and "bilinear" were proposed to increase of the prediction precision [9],[10]. However, the selection of a suitable nonlinear model and the computation of its parameters are difficult tasks for a practical problem especially when the time series behavior is non-deterministic. Moreover, it has been shown that the capability of the nonlinear model is limited, because it is unable to provide a long-term prediction [11].

In recent years, artificial intelligence tools have been extensively used for time-series based prediction [12, 13]. In particular, artificial neural networks are frequently exploited for time-series based prediction of systems behavior. A neural network is an information processing system that is capable of treating complex problems of pattern recognition, dynamic and nonlinear processes. In particular, it can be an efficient tool for prediction applications. The advantage of neural networks based approaches over statistical ones is the capability of learning and accordingly generalization of their knowledge [14]. Also the neural networks are based on training and in many cases their prediction results are more precise, even if the training set has considerable noise [14]. These approaches are much more suitable for real world problems which *do not have specific rules*.

There are some composite approaches which try to take the advantage of the accuracy of statistical models and the generality of neural network approaches. In [15], authors composed statistical model ARIMA and *a feed-forward neural network* to forecast time series. A feed forward network is a type of neural networks where all of its connections have the same direction [16]. This composition could be efficient in predicting some *well-known* time series. However, in the case of other time series, finding the proper value for statistical part of the composition is a difficult task and wrong values could affect the accuracy of prediction. Also it has been proved that the capability of *recurrent neural networks* is equivalent to the Turing machine [17]. Recurrent network is a class of neural network where connections between the layers of it could be backward or forward [16]. Therefore recurrent networks can approximate any function by learning from the function inputs and outputs.

3. Potential deadlock prediction

In our previous work we proposed an online predictive model to detect the potential deadlocks in multithread programs which is the basis of our approach [4]. Figure 1 shows our proposed model architecture. This model is consists of four components which are collaborating together at runtime. In this work, we aim to extend the basis model, indeed we extend "predictor" component, to be able to generate much more accurate prediction.

3.1 The basis of proposed model

We used dependency graph in our model which nodes are the concurrent threads or processes. There is an edge from node A to node B if and only if thread A wants to acquire a lock which held by thread B and A has to wait until B release the lock, after that the edge will be erased. There is a deadlock in the system if there is a cycle in dependency graph. Therefore, except requesting or releasing the locks, other behavior of threads does not play any role in deadlock occurrence. For this reason in our proposed model we target only the instructions which are related to acquiring or releasing the locks. We named this type of instructions deadlock-prone behavior. The main difference between our approach and other online potential deadlock detection approaches which we explained in Section 2, is that we try to predict the future deadlock-prone behavior of threads at runtime rather than try to abstract different execution traces from the current execution by changing threads schedules or noise injection. If we could have an accurate view of future deadlock-prone behavior of threads then we can accurately result about the deadlock occurrence in the future [4].

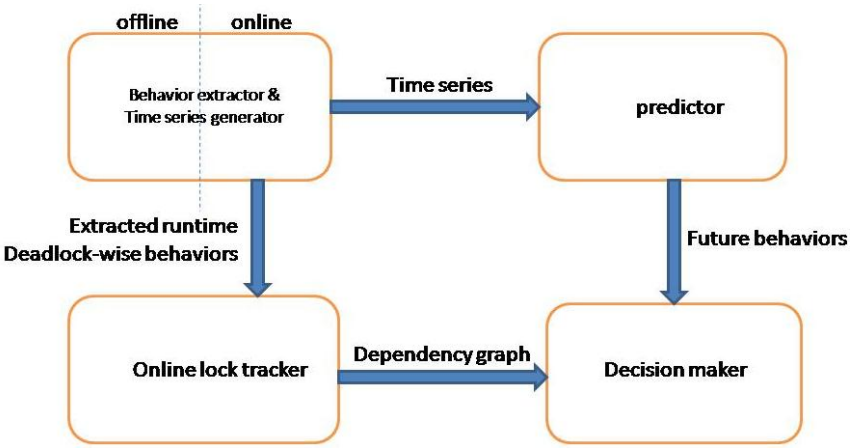


Fig 1. The basis of proposed model [4]

The start point of our model is the "Behavior extractor & Time series generator" component. Actually this component is composed of two elements:

Two annotated Java functions: one for extracting deadlock-prone behavior and another for converting extracted behavior to univariate time series. Figure 2 shows these two functions: *1- extractor & convertor () 2- this Period behaviors ()*. The first one task is catching lock () and unlock () at runtime and the second one task, is appending these instructions to the proper time series.

```
1. @AfterRunning( pointcut = "execution(*
   java.util.concurrent.locks.unlock(..))")
2. @Before(pointcut = "execution(*
   java.util.concurrent.locks.lock(..))")
3. public void extarctor&convertor
   (JoinPoint joinPoint) {
4. String
   functionName=joinPoint.getSignature().ge
   tName();
5. If(functionName.eguals("lock")){
6. thisPeriodBehaviors("1",
   Arrays.toString(joinPoint.getArgs()),this.n
   ame);
7. }
8. Else{
9. thisPeriodbehaviors("2",
   Arrays.toString(joinPoint.getArgs()),this.n
   ame);
10. }
}
```

Fig 2. Functions pseudo code

Line 1, shows an annotation which means: whenever an Unlock() instruction executed, the *extractor&convertor(…)* method, should be executed immediately. Line 2, shows an annotation which means: right before the execution of a Lock() instruction, the *extractor&convertor(…)* method, should be executed. Line 3 is the method sign and line 4, is for obtaining the name of the event which caused the *extractor&convertor(…)* method to be executed. In line 5 to 10, based on the name of event (lock or unlock), a specific character will be appended to a specific time series. In this way all the lock() and unlock() events which are

issued from threads at runtime, are caught and converted to time series.

ApectJ compiler. The task of this compiler is weaving two Java functions to the target multithread programs in the locations which are specified by annotations above the method.

We mentioned that, one of the problems in runtime verification approaches is source code instrumentation step. The instrumentation is a time consuming task and when the verification logic is complex, it could be inefficient at runtime, both in time and space. But our two Java functions which are weaved to the target multithread program, are easy and light weight thus their runtime overhead is negligible.

As we said, there are two Java functions for extracting dedicated behavior and time series generating goals. Time series is a set of observations from past until present, denoted by s(t-i) {i= 0.. P }, where P is the number of observations. Time series prediction is to estimate future observations, let's say s (t+i) for {i= 1.. N}, where N is the size of prediction window. Also, a univariate time series refers to the set of values over the time of a single quantity.

The next component in our model is "Online Lock Tracker". According to Figure 1, this component takes the deadlock-prone behavior from "Behavior extractor & Time series generator" component at runtime and draws a dependency graph. This dependency graph will be updated whenever a thread issues a deadlock-prone behavior.

The "predictor" component takes the generated time series from "Behavior extractor & Time series generator" and tries to predict the next members of the time series. In a multithread program, the order of executed instructions of a thread could be affected by other threads executions. This fact makes the concurrent systems nondeterministic thus it is hard to predict the future thread behavior. We can't assume any pre-defined generator for the time series which are representing threads behavior. This property makes the statistical prediction techniques useless for our purpose. Because the statistical prediction techniques, assume that a time series is generated by linear or nonlinear process, but the selection of the suitable nonlinear or linear model and computation of its parameters is a difficult task for a practical problem without a priori knowledge about the time series[10]. The prediction requirements of our model lead us to use artificial intelligence prediction techniques. Time series prediction techniques which are based on AI use several Artificial Neural Networks [10]. Based on the properties of time series, there are different network topologies and learning algorithms. The selection of a proper network model and adjustment of its parameters should be carried out by considering the problem requirements.

The predictions of the "predictor" component are also in the form of time series. These predictions and current dependency graph (the output of "online lock tracker" component) are injected to the "Decision maker" component. This component is responsible for deciding about the deadlock possibility in the next period. We try to clarify our model using an example. Assume that we have four threads named T1,…,T4 and five locks named L1, …,L5. Also assume the current dependency graph is something like Figure 3 (a). This graph represents that T1 has held L1 and L3 and wants to hold L2 which held by T2 thus T1 stops proceeding and waits until T2 releases L2. Also T4 has held L5 and wants to hold L3 which held by T1 thus T4 stops proceeding. Suppose the predictions of "Predictor" component are the following:

"Predictor" component predictions → T3={ will request L5}, T2={ will request L4}

"Decision maker" takes current lock graph and predictions and composes them together to construct an abstract graph. Afterwards, decision maker searches the abstract graph to find a cycle. If so, it reports a possibility of deadlock in the next period. Figure 3 (b) shows the abstract graph of our example as a composition of predictions and dependency graph.

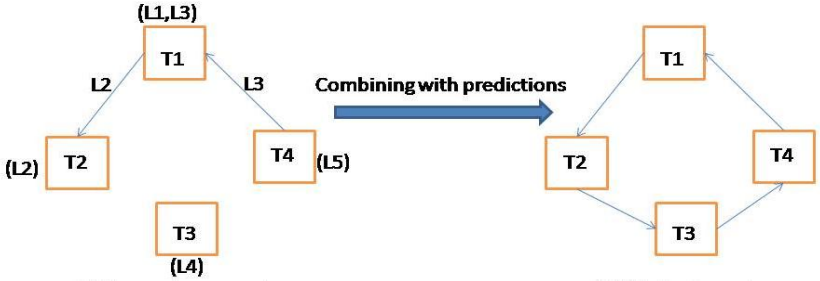


Fig 3 (a). Current lock graph

Fig 3(b). Resulted abstract graph

In our example, the abstract graph has a cycle, therefore the "Decision maker" component reports: (1) a deadlock possibility in the next period and (2) T1 to T4 as the threads will be involved in this deadlock. But, if the predictions are:

"Predictor" component predictions → T3={will request L5}, T2={ will request L4 and will release L2}

For this case, Figure 4 shows the abstract graph where there is not any cycle. Therefore, the "Decision maker" component will not report any possibility of deadlock in the next period.

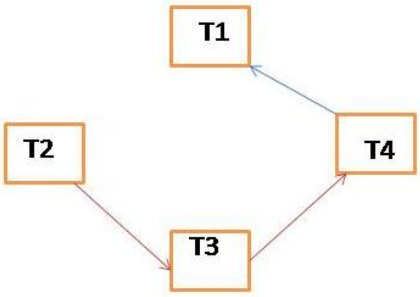


Fig 4. Resulted another abstract graph

3.2 Applying the Extensions

In our previous work we used a recurrent neural network named non-linear autoregressive (NAR) in predictor component. A NAR network tries to predict the future element of a given time series using d last values of that series [18]. That is, NAR network assumes the future element of a series is a function of its last values (Formula 1).

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-d)) \quad (1)$$

The structure of NAR network has been shown in Figure 5. This network has d inputs, each for one of the last values of time series.

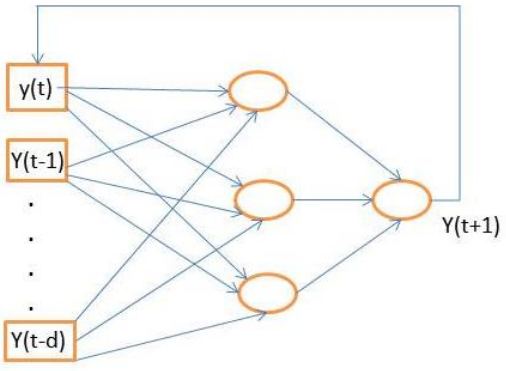


Fig 5. Structure of a NAR network

We named d as the delay parameter and it is one of the important factors which directly imposes the precision of predictions in a predictor neural network. Suppose in a time series each element is dependent on two last elements, That is $y(t) = f(y(t-1), y(t-2))$. If we try to predict $y(t)$ using a predictor neural network such as NAR, the most accurate results will be acquired if $\text{delay} = 2$. Actually in this way the network considers two last elements in predicting the future element. In previous work we obtained the proper value for delay parameter using "try & fail" approach. That is, we gathered the runtime behavior of our multithread test program and converted them into the time series. Then we tried to predict the future members of test time series, using multiple NAR networks so that every network had a different value for delay parameter from others. After that we chose the delay value of a network which made the most precise predictions.

In this work we improve the prediction precision of our "predictor" component, by configuring the delay parameter of network using a time series analysis methods. "Embedded dimension" is a factor which determines the relationships among the past and future members of a time series [19]. The value of the "Embedded dimension" for a time series represents the optimum number of last elements which every element is dependent on. Therefore we apply the "Embedded dimension "as the delay parameter in our predictor network. To obtain the "Embedded dimension" of a time series there are multiple approaches. The most known approach is False-Nearest-Neighbor, algorithm. This algorithm was firstly proposed by Kennel et al [20]. The calculation of the "Embedded dimension" allows one to extract the process behavior parameters from the observed series of events [19]. The predictor network can be further configured according to the obtained results from False-Nearest-Neighbor (FNN), in order to remember the required number of last elements in time series.

In this work, in addition to applying "Embedded dimension" as the delay parameter, we use "Nonlinear Autoregressive with External input" (NARX) network instead of NAR network. Because in our model the major task of the predictor network is predicting threads behavior at runtime. But the behavior of threads is not completely separate from each other, actually the future behavior of each thread is affected by other threads past and current behavior. Thus we need a prediction methodology which could satisfy this requirement. As it is obvious, the NAR does not consider an external input in its prediction procedure. Because of this limitation of NAR, it may not meet our prediction requirements properly. We need a prediction method which could consider other series (that is,

other thread's behavior) in predicting a time series.

NARX network, like NAR network, is a recurrent network with an external input [21]. The main idea of recurrent networks is providing a weighted feedback connection between layers of neurons and adding time significance to entire network. Therefore, recurrent neural networks simulate a temporal memory and are suitable for tasks like prediction which need a memory for the past events. NARX network assumes the future element of a given time series is a function of its last elements and another series last elements (Formula 2).

$$y(t+1) = f\big(y(t), y(t-1), \ldots, y(t-d), x(t), x(t-1), \ldots, x(t-k)\big) \quad (2)$$

Using this external input, it is possible to predict a time series considering the last elements of the time series under prediction and also considering other time series last elements. Figure 6 shows our extended "Predictor" component.

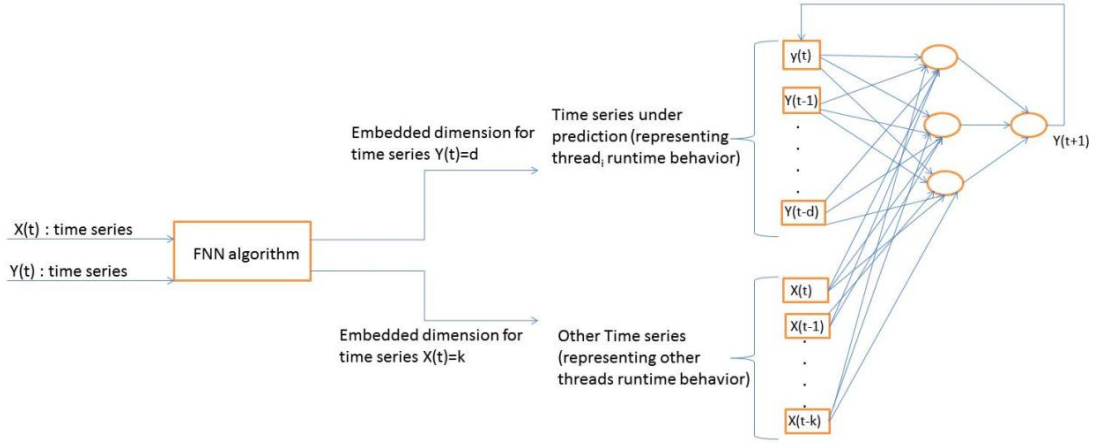


Fig 6. The extended "Predictor" component

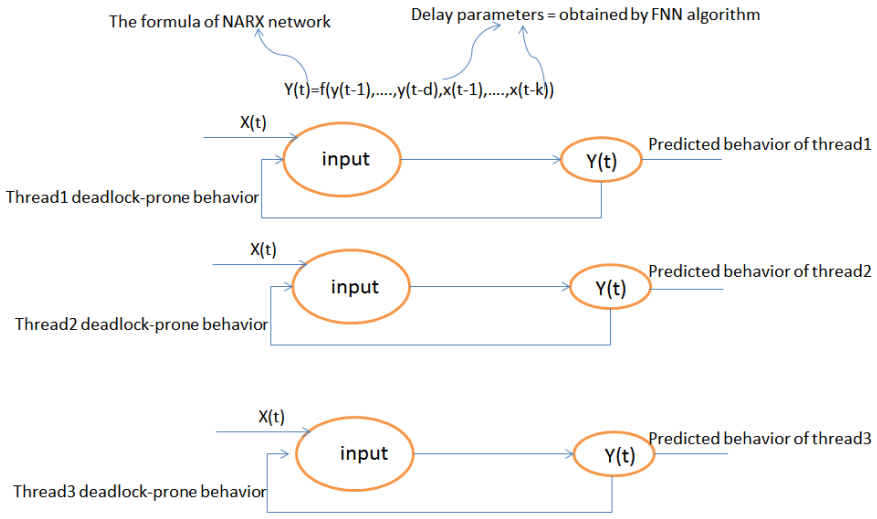


Fig 7. The NARX networks of example

To clarify the differences between the previous and current "Predictor" component at runtime, we use an example. Suppose there are three time series at runtime, then the "predictor" component will have three networks, each for predicting one of the series future elements. Each network uses some last members of target time series named y(t), and some last members of the other series named x(t), as its inputs. Therefore the new predictor networks have been shown in Figure 7, but the networks of our "previous" predictor component have been shown in Figure 8. It is obvious from the Figure 7 that, in the "predictor" component there are three NARX networks, each for predicting one of the threads (time series) future behavior. The output of a NARX network is a function of its two inputs named x(t) and y(t), therefore each network takes a target time series last behavior and another time series which represents the last behavior of the other threads. Future behavior of y(t) predicted by its past behavior and also the past behavior of x(t) and the number of last behavior obtained by FNN algorithm.

But in our previous work for this example, we there were three NAR networks and Future behavior of y(t) predicted by only its past behavior and the number of last behavior obtained by "try & fail " approach.

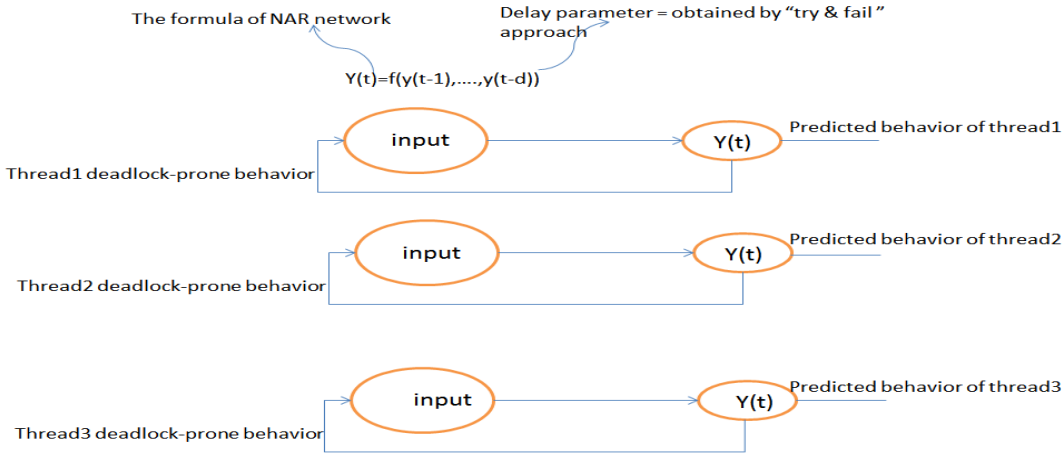


Fig 8. The NAR networks of example

4. Evaluation of the results

4.1 Experiments

Our model needs a preparation phase before that it could be used at runtime. This phase is related to configuring and training the predictor networks. For this reason first of all we should run the target multithread program for a while and gather the generated time series by "Behavior extractor & Time series generator" component during these test runs. We named these time series training phase information. Therefore we have to apply this information to train the networks and to measure the embedded dimension of time series using False-nearest-neighbor algorithm. Afterwards the obtained embedded dimensions should be used as the delay parameters in the networks. After this phase our model is ready to be used at runtime. We tested our proposed model using a Java written multithread program which consists of 50 threads and 10 shared locks. We will refer to the test multithread program as the target program in the remaining of this paper. We ran the target program 100, 500 and 1000 times. We measured and divided the failure rate in predicting future behavior of threads in four categories:

1. Failure rate based on our previous work [4] (which we: (1) considered no embedded dimension as the delay parameter and (2) did not count the other threads behavior in predicting each thread behavior)
2. Failure rate when we count the other threads behavior in predicting each thread behavior
3. Failure rate when we include embedded dimension as the delay parameter
4. Failure rate when we: (1) include embedded dimension as the delay parameter and (2) count the other threads behavior in predicting each thread behavior

Each category was considered using different trains, validations and test sets. Tables 1 to 4 show results using Markov Chain where 15%, 20%, 30% and 40% of data were respectively used for testing and 85%, 80%, 70% and 60% of data were respectively used for validating and training the networks. Similarly, Tables 5 to 8 show results using NARX model where 15%, 20%, 30% and 40% of data were respectively used for testing and 85%, 80%, 70% and 60% of data were respectively used for validating and training the networks.

Table 1. Failure rate using markov chain with 15% test data and 85% validation and train data

Runs	Failure Rate	Test Data Percentage	Validation Data Percentage	Train Data Percentage
100	-2.16	15%	15%	70%
500	-2.3	15%	15%	70%
1000	-2.5	15%	15%	70%

Table 2. Failure rate using Markov Chain with 20% test data and 80% validation and train data

Runs	Failure Rate	Test Data Percentage	Validation Data Percentage	Train Data Percentage
100	-2.2	20%	15%	65%
500	-2.45	20%	15%	65%
1000	-2.36	20%	15%	65%

Table 3. Failure rate using Markov Chain with 30% test data and 70% validation and train data

Runs	Failure Rate	Test Data Percentage	Validation Data Percentage	Train Data Percentage
100	-2.39	30%	15%	55%
500	-2.49	30%	15%	55%
1000	-2.62	30%	15%	55%

Table 4. Failure rate using Markov Chain with 40% test data and 60% validation and train data

Runs	Failure Rate	Test Data Percentage	Validation Data Percentage	Train Data Percentage
100	-3.59	40%	15%	45%
500	-3.89	40%	15%	45%
1000	-3.87	40%	15%	45%

Table 5. Failure Rate using NARX with 15% test data and 85% validation and train data

Runs	Embedding Dimension	Environment Conditions	Test Data Percentage	Validation Data Percentage	Train Data Percentage	Failure Rate
100	NO	NO	15%	15%	70%	6.119e-1
100	YES	NO	15%	15%	70%	6.054e-1
100	NO	YES	15%	15%	70%	6.043e-1
100	YES	YES	15%	15%	70%	5.801e-1
500	NO	NO	15%	15%	70%	8.719e-1
500	YES	NO	15%	15%	70%	4.57e-1
500	NO	YES	15%	15%	70%	5.962e-1
500	YES	YES	15%	15%	70%	4.411e-2
1000	NO	NO	15%	15%	70%	8.212e-1
1000	YES	NO	15%	15%	70%	4.008e-1
1000	NO	YES	15%	15%	70%	6.009e-1
1000	YES	YES	15%	15%	70%	3.089e-1

Table 6. Failure Rate using NARX with 20% test data and 80% validation and train data

Runs	Embedding Dimension	Environment Conditions	Test Data Percentage	Validation Data Percentage	Train Data Percentage	Failure Rate
100	NO	NO	20%	15%	65%	6.093e-1
100	YES	NO	20%	15%	65%	6.043e-1
100	NO	YES	20%	15%	65%	6.085e-1
100	YES	YES	20%	15%	65%	5.221e-1
500	NO	NO	20%	15%	65%	8.332e-1
500	YES	NO	20%	15%	65%	4.431e-1
500	NO	YES	20%	15%	65%	5.101e-1
500	YES	YES	20%	15%	65%	4.01e-2
1000	NO	NO	20%	15%	65%	8.77e-1
1000	YES	NO	20%	15%	65%	3.981e-1
1000	NO	YES	20%	15%	65%	6.764e-1
1000	YES	YES	20%	15%	65%	3.821e-1

Table 7. Failure Rate using NARX with 30% test data and 70% validation and train data

Runs	Embedding Dimension	Environment Conditions	Test Data Percentage	Validation Data Percentage	Train Data Percentage	Failure Rate
100	NO	NO	30%	15%	55%	7.498e-1
100	YES	NO	30%	15%	55%	6.327e-1
100	NO	YES	30%	15%	55%	6.59e-1
100	YES	YES	30%	15%	55%	6.481e-1
500	NO	NO	30%	15%	55%	10.112e-1
500	YES	NO	30%	15%	55%	6.001e-1
500	NO	YES	30%	15%	55%	6.439e-1
500	YES	YES	30%	15%	55%	5.021e-1
1000	NO	NO	30%	15%	55%	9.114e-1
1000	YES	NO	30%	15%	55%	5.11e-1
1000	NO	YES	30%	15%	55%	7.872e-1
1000	YES	YES	30%	15%	55%	5.082e-1

Table 8. Failure Rate using NARX with 40% test data and 60% validation and train data

Runs	Embedded Dimension	Environment Conditions	Test Data Percentage	Validation Data Percentage	Train Data Percentage	Failure Rate	Average Failure
100	NO	NO	40%	15%	45%	13.309e-1	8.61E-01
100	YES	NO	40%	15%	45%	7.006e-1	
100	NO	YES	40%	15%	45%	8.12e-1	
100	YES	YES	40%	15%	45%	6.006e-1	
500	NO	NO	40%	15%	45%	14.589e-1	7.99E-01
500	YES	NO	40%	15%	45%	5.043e-1	
500	NO	YES	40%	15%	45%	8.229e-1	
500	YES	YES	40%	15%	45%	4.1e-1	
1000	NO	NO	40%	15%	45%	12.984e-1	4.90E-01
1000	YES	NO	40%	15%	45%	9.034e-2	
1000	NO	YES	40%	15%	45%	5.002e-1	
1000	YES	YES	40%	15%	45%	7.001e-2	

The 1st, 5th and 9th rows from every NARX table show the results of prediction based on our previous work. The failure rate of the rows which consider the extensions is much more accurate. Therefore we can say, importing the new extensions in this work, that is, embedded dimension as the delay parameter and considering each thread behavior in predicting other threads future behavior, made considerable improvement in prediction results particularly when the number of runs increases. We also showed the prediction results of NARX networks was much more accurate than the results obtained by Markov Chain, which is a statistical approach. As we stated, our test target program behaves randomly at runtime. Therefore, it was not possible to suppose an accurate model for Markov Chain prediction strategy. This is why that the failure rate of this strategy, as shown in Tables 1 to 4, are imprecise in comparison with the similar tables of the NARX prediction.

The average results of every NARX table (Figure 9) show a comparative view of the results of this strategy. Every line marked with a (X,Y,Z) statement, which X means the test set percentage, Y means validation set percentage and Z means the training set percentage. When the training set percentage is significantly lower than twofold test set percentage, the failure rate will increase. Also as the number of runs increases the effect of training is much more visible. According to the chart, the best overall result is in the case of (20, 15 and 65). This result is dedicated for our target multithread program and it may differ for other multithread programs.

In [4], after training networks we ran target program 500 times and tried to predict the deadlock possibility during these runs. During these runs deadlock occurred 17 times. Our approach reported 13 before their occurrences and missed 4. Also in 3 cases, it reported false positive, thus the precision was about 74%. In this work, after training the networks using considered extensions, we again ran test multithread program 500 times to see how many deadlocks will be reported correctly. It results 15 deadlocks during 500 times. Our model, this time, reported 14 and missed just one deadlock not reported; also it didn't report any false positive. This time, the precision was about 88%. In

comparison with our previous work [4], the extensions made a clear improvement in the results up to 15%.

5. Conclusion

Online potential deadlock detection techniques received lots of attention in recent years. But these techniques often are not cost efficient, neither in time or space. Also they need extra programmer effort to instrument the code and in some cases the results of these techniques may come too late. Considering these problems we proposed a novel online model to predict the deadlock at runtime in multithread programs rather than discovering deadlocks by pre-running some execution traces to find the potential deadlocks. In our proposed model the main runtime overhead is through the predictor component which predicts the future behavior of threads using neural network. In this work we used the "Nonlinear Autoregressive with external input (NARX)" network. The learning phase of NARX network has the order of complexity $O(n^3)$ in worst case [22]. But this complexity is related to offline phase of our proposed model and once the networks were trained, then at runtime the output of predictor will be generated with a lower order of complexity, therefore our model doesn‘t force considerable overhead at runtime. Also our model could be execute on a completely different core from the main program and because of the simplicity of instrumentation logic it doesn‘t interfere in the target program execution.

In this work we extended our previous work in two ways:

1. Using time series analysis approaches in configuring predictor network parameter
2. Using NARX network instead of NAR network.

The obtained results showed that the extensions described in this paper, made improvement in the prediction of potential deadlocks. The configuring a predictor neural network considering the problem specification and requirements resulted the more precise predictions. Because of this experience, in our future work, we are planning to configure the predictor networks parameters based on the static analysis and structure of the target multithread program, we hope to obtain more accurate results.

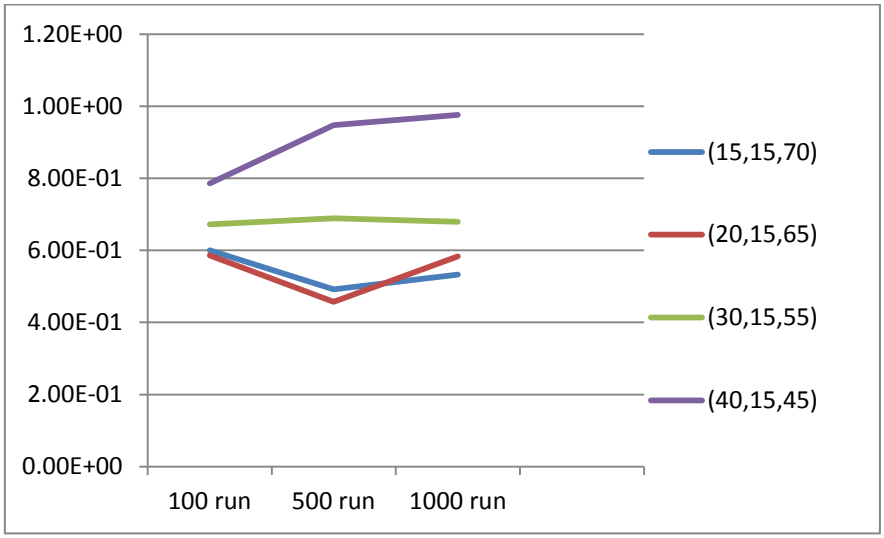


Fig. 9. Average results of NARX strategy with different test, validation and train data

References

[1] D. Engler, K. Ashcraft,“ RacerX: effective, static detection of race conditions and deadlocks,” ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 237–252, 2003.

[2] P. Joshi, C. S. Park, K. Sen, and M. Naik, “ A randomized dynamic program analysis technique for detecting real deadlocks,” in ACM Sigplan Notices, 2009, vol. 44, pp. 110–120.

[3] H. Jula, D. Tralamazza, C. Zamfir, G. Candea, “ Deadlock immunity: Enabling systems to defend against deadlocks,” in Proceedings of the 8th USENIX conference on Operating systems design and implementation, 2008, pp. 295–308.

[4] E. Hasanzade , S. M. Babamir, “An Artificial Neural Network Based Model for Online Prediction of Potential Deadlock in Multithread

Programs," in 16th Symposium of Artificial Intelligence and Signal Processing, AISP 2012, IEEE Society, pp. 417-422, 2012.

[5] Klaus Havelund. Using runtime analysis to guide model checking of java programs. In Proc. 7th Int'l. SPIN Workshop on Model Checking of Software,volume 1885 of Lecture Notes in Computer Science, pages 245-264. Springer-Verlag, August 2000.

[6] R. Agarwal, L. Wang, S. Stoller, " Detecting potential deadlocks with static analysis and run-time monitoring," Hardware and Software, Verification and Testing, pp. 191–207, 2006.

[7] S. Bensalem, K. Havelund, "Scalable deadlock analysis of multi-threaded programs," in Proceedings of the Parallel and Distributed Systems: Testing and Debugging (PADTAD) Track of the 2005 IBM Verification Conference. Springer-Verlag, 2005.

[8] Y. Nir-Buchbinder, R. Tzoref, S. Ur, "Deadlocks: From exhibiting to healing," in Runtime Verification, 2008, pp. 104–118.

[9] O. Voitcu, Y. Wong, "On the construction of a nonlinear recursive predictor," Science B.V., Journal of Computational and Applied Mathematics, 2004.

[10] N. Baccour, H. Kaaniche, M. Chtourou, M. B. Jemaa, " Recurrent neural network based time series prediction: Particular design problems," studies, vol. 1, p. 7.

[11] Y. Chen B. Yang J. Dong A. Abraham, " Time-series forecasting using flexible neural tree model," Science, Information Sciences pp 219–235, 2004.

[12] C.J. Lin, Y.J. Xu, "A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications," Science, Fuzzy Sets and Systems, 2 September 2005.

[13] R. Zemouri, P. Ciprian Patic "Recurrent Radial Basis Function Network for Failure Time Series Prediction," World Academy of Science, Engineering and Technology 72, 2010.

[14] R. Zemouri, D. Racoceanu, N. Zerhouni, "Recurrent radial basis function network for time-series prediction," Engineering Applications of Artificial Intelligence pp. 453–463, 2003.

[15] M. Khashei, M. Bijari, " An artificial neural network (p,d,q) model for time series forecasting" Journal of Expert Systems with Applications, pp. 479-489, 2010.

[16] R. Rojas, "Neural networks: a systematic introduction," Springer. pp. 336, 1996.

[17] H. Hyotyniemi**, "** Turing Machines are Recurrent Neural Networks," Proceedings of STeP'96. Jarmo Alander, Timo Honkela and Matti Jakobsson, pp. 13-24, 1996.

[18] G. Dorffner, "Neural Networks for Time Series Processing," Neural Network World, Vol. 6, No. 4, 447-468, 1996

[19] E. Dodonov, R. F. de Mello, "A novel approach for distributed application scheduling based on prediction of communication events," Future Generation Computer Systems, vol. 26, no. 5, pp. 740–752, 2010.

[20] M.B. Kennel, R. Brown, H.D.I. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction", Physical Review A 45 (6) (1992) 34033411.

[21] T.Lin, B.G. Horne, P.Tino, C. Lee Giles, " Learning long-term dependencies in NARX recurrent neural networks," IEEE Transactions on Neural Networks, Vol. 7, No. 6, 1996, pp. 1329-1351

[22] G. Ferrari, G. De Nicolao, "NARX models: optimal parametric approximation of nonparametric estimators," in American Control Conference, 2001. Proceedings of the 2001, 2001, vol. 6, pp. 4868–4873.

Network RAM Based Process Migration for HPC Clusters

Hamid Sharifian*
Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
sharifian@comp.iust.ac.ir
Mohsen Sharifi
Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
msharifi@iust.ac.ir

Received: 04/Dec/2012 Accepted: 09/Feb/2013

Abstract
Process migration is critical to dynamic balancing of workloads on cluster nodes in any high performance computing cluster to achieve high overall throughput and performance. Most existing process migration mechanisms are however unsuccessful in achieving this goal proper because they either allow once-only migration of processes or have complex implementations of address space transfer that degrade process migration performance. We propose a new process migration mechanism for HPC clusters that allows multiple migrations of each process by using the network RAM feature of clusters to transfer the address spaces of processes upon their multiple migrations. We show experimentally that the superiority of our proposed mechanism in attaining higher performance compared to existing comparable mechanisms is due to effective management of residual data dependencies.

Keywords: High Performance Computing (HPC) Clusters, Process Migration, Network RAM, Load Balancing, Address Space Transfer.

1. Introduction

A standard approach to reducing the runtime of any high performance scientific computing application on a high performance computing (HPC) cluster is to partition the application into several portions that can be run in parallel by multiple cluster nodes simultaneously.

HPC clusters generally consist of three main parts: a collection of off-the-shelf (COTS) computing and storage nodes, a network connecting the nodes, and a cluster manager system software that manages all nodes and presents a single system image to applications while exploiting the parallel processing power of multiple nodes.

The cluster manager system software provides a set of global services that aim at making resource distribution transparent to all applications, managing resource sharing between applications, deploying as much cluster resources as possible for demanding applications, and scheduling parallel processes on all cluster nodes. The services include global resource management, distributed scheduling, load sharing, process migration, and network RAM.

Dynamic load sharing can be achieved by moving processes from heavily-loaded nodes to lightly-loaded nodes at runtime. This can lead to fault resilience, ease of system administration, and data access locality in addition to an enhanced degree of dynamic load distribution [1].

Upon migration of a process, the process must be suspended and its context information in the source node extracted and transferred to the destination node. The process can only then resume executing from the point it was suspended. Two critical challenges of process migration are the transfer of the process address space from the source node to the destination node, and access to the opened files in the destination node after process migration [2].

In this paper we only focus on resolving the first challenge of process migration by introducing a new process migration mechanism by using the network RAM feature of HPC clusters, wherein the aggregate main memory of all cluster nodes in a cluster represents the network RAM of that cluster [3]. In addition to achieve comparably higher performance than existing process migration mechanisms, our proposed mechanism is intended to allow for multiple-migration of each process that is a missing feature in existing process migration mechanisms that is accounted as a source of inefficiency.

The rest of paper is organized as follows. Sections 2 and 3 introduce process migration and network RAM. Section 4 reviews related works on process migration with respect to transfer of

* Corresponding Author

process address space. Sections 5 and 6 present our network RAM based mechanism and its evaluation, and Section 7 concludes the paper.

2. Process Migration

Process migration is the act of transferring an active process between two computers and restoring its execution in a destination node from the point it left off in the source node. The goals of process migration are closely tied to applications that use migration. The primary goals include resource locality, resource sharing, dynamic load balancing, fault resilience, and ease of system administration [1]. Any process migration mechanism can thus be benchmarked and evaluated with respect to the degree it satisfies these goals.

Considering an HPC cluster, process migration has three main phases [4] (note that these phases are applicable to process migration in all types of networks of computers in general including HPC clusters that are the main focus of this paper):

1. *Detaching Phase* that involves the suspension and the extraction of the context of a migrant process in its current node. These activities must be done in a way that none of the other processes running on the current node or in other nodes of the cluster experience any execution inconsistencies. At the start of this phase, the execution of the migrant process is frozen.
2. *Transfer Phase* that involves the transfer of the extracted context of the migrant process to the destination node.
3. *Attaching Phase* that involves the reconstruction of the migrant process on the destination node. The reconstruction in turn involves the allocation of resources on the target node to the migrant process, informing the beneficiaries and/or brokers of the migrant process about the current executing place of the migrant process, and resuming the execution of the migrant process on the destination node from the point it left off on the source node.

The time interval between freezing a migrant process on a source node and resuming its execution on a destination node is called the freeze time representing the status wherein the migrant process is neither executing on the source nor executing on the destination node. The longer the freeze time the lower will be the performance of the process migration.

The context of a process to be migrated includes the process's running state; stack contents; processor registers; address space; heap data; and process's communication state (like open files or message channels). The whole context must be transferred to the destination node before the process can continue its execution on the destination node. The process address space is the largest part of the process context that might have hundreds of megabyte of data [5] taking longest to be transferred to the destination node. This can adversely affect the performance of process migration. Therefore, the performance of any process migration mechanism largely depends on how long it takes to transfer the context of migrant processes to destination nodes.

Various data transfer techniques have been presented in the literature that try to reduce the high cost of address space transfer [6]. A well-known technique is to transfer only parts of process address space to allow resumption of processes on destination nodes without waiting for the transfer of the whole process address space and context. Though very attractive on grounds of improving the performance of process migration, this technique leaves parts (pages) of process address space on different nodes when multiple migrations in the lifetime of process is allowed and occurs. In other words, the process address space is scattered on multiple nodes resulting in residual dependencies. Management of residual dependencies increases the implementation complexity of process migration that in turn results in performance degradation of process migration.

Our proposed mechanism is particularly useful to strong migrations wherein the entire process state (rather than code and some initialization data in weak code migration) must be transferred to destination.

3. Network RAM

Large-memory high-performance applications such as scientific computing, weather prediction simulations, data warehousing and graphic rendering applications need massive fast accessible address spaces [7] that are not provided by even high capacity DRAMs. The runtime performance of these applications degrades quickly when system encounters memory shortage and starts swapping memory to local disks.

In today's clusters with very low-latency networks, the idle memory of other nodes can be used as storage media faster than local disks, called network RAM. The goal of network RAM

is to improve the performance of memory intensive workloads by paging to idle memory over the network rather than to disk [8].

Some common uses of network RAM are remote memory paging [9], network memory file systems, and swap block devices [10,11]. Locating unused memory in every node requires that network RAM keeps up-to-date information about all unused memories.

Since network RAM stores data on remote memories, it includes remote memory paging facility to keep information on all remote data. This functionality of network RAM can be used to alleviate the performance overhead of process migration in transferring and managing the address spaces of migrant processes. This describes why we have deployed network RAM technology to propose a novel process migration mechanism for HPC clusters in this paper.

4. Related Works

We can categorize into three categories the works on process migration that have presented solutions to cope with the address space transfer issue in particular into three categories.

4.1 Address Space Transfer Techniques

To avoid the high cost of process address space transfer, several techniques have been introduced. In the *total-copy* technique, which is the simplest and weakest one, the whole process address space is copied to destination node at the migration time [12,13,14,15,16]. The *pre-copy* technique transfers the whole process address space to destination node before starting to migrate the process in order to reduce the freeze time of the process [17]. The *copy-on-reference* technique transfers only the process state to the destination node and pages of process address space are transferred on demand [18,19]. In the *flushing* technique, dirty pages are flushed to disk and the process accesses them on demand from disk instead of memory on the source node [20].

4.2 Prefetching Techniques

Besides techniques for transferring process address space, another technique is proposed to increase the performance of address space transfer while migrant is running on the destination node. This technique includes prefetching of those pages that are likely to be accessed by the migrant to avoid remote page faults. This solution is used in openMosix and it is called Lightweight Prefetching [21].

4.3 DSM-based Techniques

Some HPC clusters have used the distributed shared memory (DSM) mechanism to transfer process address spaces between nodes upon process migrations. Pages stored on DSM need not be transferred at all during process migration. Only pages accessed by the migrant after migration are provided to the migrant process using the DSM mechanisms. *Kerrighed* is a kind of SSI operating system that has used this technique for process migration [2]. *CORAL* [4] and *Mach* [6] use the same technique for process migration and *MigThread* [22] uses a DSM framework for thread migration.

4.4 Comparison

DSM-based and copy-on-reference techniques in support of process address space transfer are more efficient than their counterparts because they do not transfer the whole process address space and avoid storing pages on disk. However, systems such as *Mosix* [15], *Accent* [19] and *RHODOS* [18] that have used the copy-on-reference migration technique allow processes to migrate just once in their lifetime in order to avoid the complex implementation of multiple process migrations or better said the complex management of dependencies of data residual on different nodes if multiple migrations were allowed. On the other hand, systems that have used the DSM-based technique are quite dependent on the implementation of their DSM manager for handling dependencies between residual data on different nodes that are provided to migrant processes on demand. In this paper, we propose a transfer technique in the face of multiple process migration allowance whose performance is higher than existing implementations of the DSM-based technique.

5. Motivation

Process migration has gained popularity for several reasons. Traditional process scheduling mechanisms lack enough flexibility to cope with changing loads of very large HPC clusters and process migration can be beneficial here. Unlike other mechanisms such as check-pointing, process migration needs no server coordination [20] and is more suited to make clusters scalable. By growing the size of data in high performance applications rather than process code size which is quite stable, process migration will be very promising when data are located on several nodes.

In spite of above advantages, process migration has not been widely used in HPC

clusters. This is mainly due to the low performance of process migration mechanisms and the complexity of implementing the migration support in commodity operating systems.

In HPC clusters executing applications with huge address spaces, the use of idle memories of remote cluster nodes instead of disk is more attractive. This can be achieved by network RAM. With growing applications with large data, the use of network RAM has become more advantageous. That is why various models have been implemented in recent years [7,9,10,11,23,24]. In such applications whose address spaces are very large and distributed among multiple nodes, process migration is more beneficial because of the smaller sizes of the process states, though data distribution makes data provision to migrant processes more difficult.

By using the network RAM technology, memory is managed without any direct interference of process migration mechanism simplifying the implementation of process migration and improving the overall performance of process migration mechanism.

6. Proposed Mechanism

Network RAM uses memories of remote cluster nodes to store data. When a process migrates to another node, some parts of its address space are remained on the source node. This is similar to the case that the process is on the destination and data are stored in remote memory of the network RAM that becomes accessible to the migrant process on demand. Inspired by this similarity, we propose a new process migration mechanism that uses network RAM for the purpose of transferring process address spaces during process migrations. This approach decreases the implementation complexity of our process migration mechanism, reduces the overhead of residual data dependencies, and improves the performance of migrant processes.

Network RAM has good facilities that can be used in process migration. These facilities include remote memory pager and an efficient module to locate remote pages across an HPC cluster. Thanks to these facilities, we can transfer only the required pages and at the same time, allow processes to migrate multiple times on various nodes.

In our mechanism, there is no need for transfer of the whole process address space and pages required by the migrant process can be accessed through the network RAM remotely. Fig. 1 shows the schema of our proposed mechanism.

When a process is selected for migration, no pages are transferred. Instead, each page in address space of the migrant process is added to the data structures of the network RAM and marked as a *remote page*. Then, page faults of migrant are handled by the network RAM faster than DSM-based solutions.

The network RAM module manages all memory-related issues of process migration mechanism including the transfer of process address space during migration, management of residual dependencies and handling page faults in addition to its own work. This simplifies the implementation of process migration mechanism.

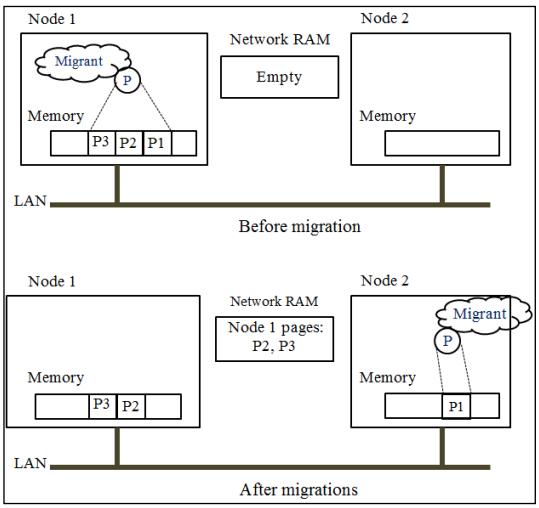


Fig.1. A schematic view of our network RAM-based process migration mechanism

7. Experimental Result

To evaluate our proposed mechanism, we simulated our mechanism by running it on a homogenous cluster with 20 nodes connected to an Ethernet switch via a 10Gbps cluster network connection. Each node had a 3GHz CPU. The cluster has global distributed shared memory, global network RAM and global process and network management features in addition to process migration facility.

We evaluated our mechanism by DGEMM HPC benchmark. Fig. 3 shows the migration times of processes with different sizes of address spaces under DSM-based, network RAM-based and copy-on-reference process migration mechanisms. Among previous solutions only DSM-based migration method support multiple migration of a process in its lifetime on different workstations.

As Fig. 3 shows, the migration times under DSM-based and our proposed mechanisms were almost equal. The migration time has increased linearly with increases in the size of process address space. In both mechanisms, the whole process address space was not transferred to the destination node at migration time. However, the larger the process address space the higher was the migration time implying that bigger address spaces take longer to be managed that is quite logical and sensible. The migration time under copy-on-reference mechanism is not dependent on address space size of process and almost remains unchanged.

So far our experiments showed the same performance for DSM-based vs. network RAM-based process migration mechanisms.

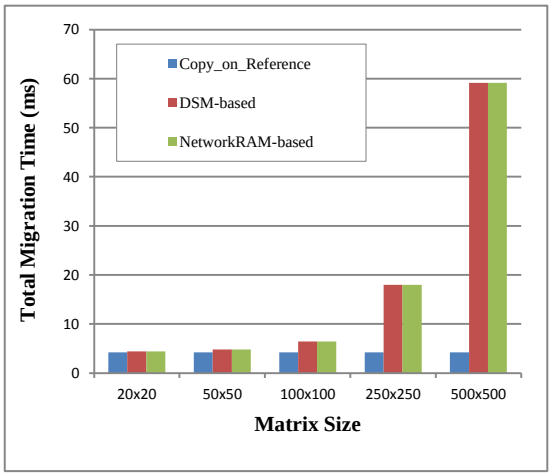


Fig.2. Total migration time for different matrix Sizes

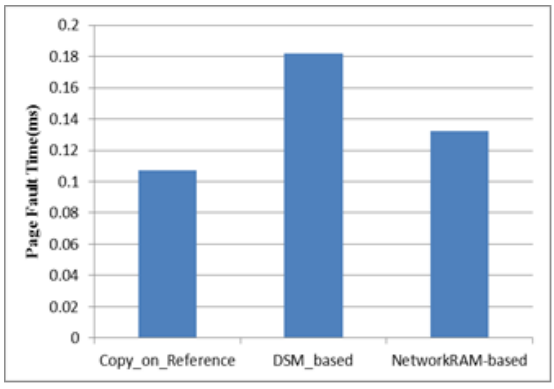


Fig.3. Page fault handle time for process migration mechanisms

The key improvement in our network RAM-based mechanism is handling page faults of the migrant process on destination node. Fig. 3 shows the average page fault handling time for DSM-based, copy-on-reference and network RAM-based mechanisms. As Fig. 3 shows, our proposed mechanism handle page fault of the migrant process faster than DSM-based mechanism. That is because network RAM-based does not consider memory sharing issues and providing pages to the demanding process is performed without locking operations. However, copy-on-reference mechanism has minimum page fault time, because in this way, requested pages are being brought from one specified workstation namely the source workstation.

Due to the fact that page faults may occur thousands of times while executing the migrant process on destination node, improvement of page fault time in our proposed mechanism results in improvement of execution time of process compared to DSM-based mechanism.

Given that our network RAM-based mechanism supports multiple migration of the process, the advantage of our network RAM-based mechanism showed even more when a process was allowed to migrate to more than one cluster node in its lifetime. The more a process is selected to migrate, the more page faults it may experience in its execution.

Fig. 4 shows the execution times of the DGEMM process with 500×500 matrix size after multiple migrations on different cluster nodes in its lifetime. As a result of effective page fault handling by the network RAM, the execution times of the migrant were reduced compared to those of DSM-based mechanism.

8. Conclusion and Future Works

In this paper, we proposed a mechanism that exploited the network RAM facility existing in clusters to transfer

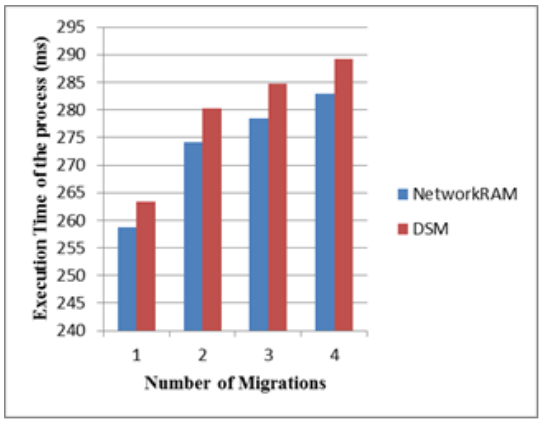


Fig.4. Execution time of DGEMM process after multiple migration in DSM- and Network RAM-based process migrations

process address spaces during process migrations in HPC clusters.

Our simulative experiments showed higher overall performance of migrant processes under our proposed mechanism compared to a simulated DSM-based process migration mechanism when processes were allowed to migrate to more than one cluster node in their life-time. This implies that our proposed mechanism is especially attractive to scientific applications with coarse-grain long-lived processes that may require multiple migrations in their life-time; we know as a fact that migration of short-lived processes is not efficient [6]. The network RAM facility we used in our proposed process migration mechanism managed access to remote memory and consequently simplified the implementation of our mechanism.

We can further improve the performance of our proposed process migration mechanism by reducing the numbers of required page transfers by coordinating the network RAM as to where to store remote pages with the task that selects a node as destination upon migration.

Acknowledgments

We hereby acknowledge the help of Mr. Reza Azariun in drafting this paper. We also wish to thank Mr. Ehsan Mousavi and Ms Mirtaheri for their guidance in initiating research on migration in HPC clusters. We also thank ITRC for partially supporting the research whose results are partially reported in this paper

.

References

[1] Nalini Vasudevan and Prasanna Venkatesh, "Design and Implementation of a Process Migration System for the Linux Environment," 3rd International Conference on Neural, Parallel and Scientific Computation, August 2006, pp. 1 - 8.

[2] Geoffroy Vall´ee, Christine Morin, Jean-Yves Berthou, Ivan Dutka Malen, and Renaud Lottiaux, "Process Migration based on Gobelins Distributed Shared Memory," Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, May 2002, pp. 325 - 325

[3] Michail D. Flouris and Evangelos P. Markatos, "Network RAM," High Performance Cluster Computing, Architectures and Systems.: Prentice Hall, vol. 1, ch. 16, pp. 383-408, 1999.

[4] Ivan Zoraja, Arndt Bode, and Vaidy Sunderam, "A Framework for Process Migration in Software DSM Environments," Proceedings of 8th Euromicro Workshop on Parallel and Distributed Processing, 2000, pp. 158 - 165.

[5] Ehsan Mousavi Khaneghah, Najmeh Osouli Nezhad, Seyedeh Leili Mirtaheri, Mohsen Sharifi, and Ashakan Shirpour, "An Efficient Live Process Migration Approach for High Performance Cluster Computing Systems," Communications in Computer and Information Science, 2011, vol. 241 part 8, pp. 362 - 373.

[6] Dejan S. Milojicic, Fred Douglis, Yves Paindaveine, Richard Wheeler, and Songnian Zhou, "Process Migration," ACM Computing Surveys (CSUR), vol. 32, no. 3, September 2000, pp. 241 - 299.

[7] Michael R. Hines, Mark Lewandowski, and Katrik Gopalan, "Anemone: Adaptive Network Memory Engine," Proceedings of the twentieth ACM symposium on Operating systems principles, 2005.

[8] Eric A. Anderson and Jeanna M. Neefe, "An Exploration of Network RAM", University of California at Berkeley, 1999.

[9] Hiroko Midorikawa, Motoyoshi Kurokawa, Ryutaro Himeno, and Mitsuhisa Sato, "DLM: A Distributed Large Memory System using Remote Memory Swapping over Cluster Nodes," Proceedings of 2008 IEEE International Conference on Cluster Computing, 2008, pp. 268 - 273.

[10] Hui Jin, Xian-He Sun, Yong Chen, and Tao Ke, "REMEM: Rmote Memory as Checkpointing Storage," 2nd IEEE International Conference on Cloud Computing Technology and Science, 2010.

[11] Changgyoo Park, Shin-gyu Kim, Hyuck Han, Hyeonsang Eom, and Heon Y. Yeom, "Design and Evaluation of Remote Memory Disk Cache," Proceedings of 2010 IEEE International Conference on Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 20-24 Sept. 2010, pp. 1 - 4.

[12] Michael L. Powell and Barton P. Miller, "Process Migration in Demos/MP," Proceedings of the ninth ACM symposium on Operating systems principles, 1983, New York, NY, USA, pp. 110 - 119.

[13] Yeshayahu Artsy and Raphael Finkel, "Designing a Process Migration Facility: The Charlotte Experience," IEEE Computer, vol. 22, no. 9, September 1989, pp. 47 - 56.

[14] Chris Steketee, Piotr Socko, and Bartosz Kiepuszewski, "Experiences with the Implementation of a Process Migration Mechanism for Amoeba," Proceedings of the 19th ACSC Conference, January-February 1996, Melbourne, Australia, p. 140—148.

[15] Amnon Barak, Oren Laden, and Yuval Yarom, "The NOW MOSIX and its Preemptive Process Migration Scheme," Bulletin of the IEEE Technical Committee on Operating Systems and

Application Environments (TCOS), vol. 7, no. 2, Summer 1995, pp. 5 - 11.

[16] Gerald Popek and B. WALKER, The Locus Distributed System Architecture: MIT Press, 1985.

[17] Marvin M. Theimer, Keith A. Lantz, and David R. Cheriton, "Preemptable Remote Execution Facilities for the V-System", Proceedings of the 10th ACM symposium on Operating systems principles, 1985, New York, pp. 2 - 12.

[18] Damien De Paoli and Andrzej Goscinski, "Copy on Reference Process Migration in RHODOS," 1996 IEEE Second International Conference on Algorithms and Architectures for Parallel Processing(ICAPP 96), Jun 1996, pp. 100 - 107.

[19] Edward R. Zayas, "Attacking the Process Migration Bottleneck," Proceedings of the 11th ACM Symposium on Operating systems principles, 1987, New York, USA, pp. 13 - 24.

[20] Fred Douglis and John Ousterhout, "Transparent Process Migration: Design Alternatives and the Sprite Implementation," Software - Practice and Experience, vol. 21, no. 8, August 1991, pp. 757 – 785.

[21] Roy S.C. Ho, Cho-Li Wang, and C.M. Francis Lau, "Lightweight Process Migration and Memory Prefetching in openMosix," IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008) , April 2008, Hong Kong, pp. 1 - 12.

[22] Hai Jiang and Vipin Chaudhary, "MigThread: Thread Migration in DSM Systems," Proceedings of International Conference on Parallel Processing Workshops, 2002, pp. 581 - 588.

[23] Nan Wang et al., "Collaborative Memory Pool in Cluster System," IEEE International Conference on Parallel Processing, 2007, Boston MA, USA, pp. 17 - 24.

[24] Paul Werstein, Xiangfei Jia, and Zhiyi Huang, "A Remote Memory Swapping System for Cluster Computers," Proceedings of Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies, 3-6 Dec. 2007, pp. 75 - 81.

Accurate Fire Detection System for Various Environments using Gaussian Mixture Model and HSV Space

Khosro Rezaee*
Department of Electrical and Computer Engineering, Hakim Sabzevari University, Iran
rezaeekhosro@ymail.com
S. Jalal Mousavirad
Department of Computer and Information Technology, University of Kurdestan, Iran
jalalmoosavirad@gmail.com
Mohammad RaseghGhezelbash
Department of Electrical and Computer Engineering, Hakim Sabzevari University, Iran
m.rasegh@gmail.com
Javad Haddania
Department of Electrical and Computer Engineering, Hakim Sabzevari University, Iran
jhaddania@yahoo.com

Received: 23/Sep/2012 Accepted: 21/Oct/2012

Abstract
Smart and timely detection of fire can be very useful in coping with this phenomenon and its inhibition. Enhancing some image analysis methods such as converting RGB image to HSV image, smart selecting the threshold in fire separation, Gaussian mixture model, forming polygon the enclosed area resulted from edge detection and its combination with original image, this papers addresses fire detection. Accuracy and precision in performance and rapid detection of fire are among the features that distinguish this proposed system from similar fire detection systems such as Markov model, GM, DBFIR and other algorithms introduced in valid articles. The average accuracy (95%) resulted from testing 35000 frames in different fire environments and the high sensitivity (96%) was quite significant. This system be regarded as a reliable suitable alternative for the sensory set used in residential areas, but also the high speed image processing and accurate detection of fire in wide areas makes it low cost, reliable and appropriate.
Keywords: Fire Detection, Gaussian Mixture Model, Image Processing, HSV Space, Edge Detection.

1. Introduction

Fire Detection is an important issue in today's world because it directly threatens the life of living organisms, especially human life. Researches dealing with fire detection have always been important since the spread and the impact of fire detection in residential areas, business centers, industrial areas and open areas is critical. However, the field of smart fire detection is not confined to these areas. Fire detection without human intervention seems necessary in jungles, parks and farms too. The conventional sensors of fire detection such as environmental heat detector or smoke detector has a significant role in fire and smoke detection. Nevertheless, recent studies have shown that in large places and buildings, smoke detectors and heat sensors are not desirables due to point detection [1]. On the other hand, the employment of these systems in such open areas as jungles and spacious storages is expensive, having reduced capability and accuracy and unable to entirely cover these areas. In jungles, we need monitoring, employing human resources and large quantity of sensory network. Fire detection through image processing system is a new method based on one or some main color indexes, composition or formal structure and brightness. In some methods, colored mass of fire used for fire detection. Turgay et al [2] attempted to detect fire using fuzzy logic and colored mass of fire. Of course they first attempt to trace the smoke in video sequences, but they detect fire flames by using combine methods in next steps. Wirth and Zaremba [3] developed an efficient system by analyzing Back projection histogram image and properties of fire pixels in the image. Flame region is target for them in video sequences with high rate detection to

* Corresponding Author

prevention from sudden firing in apartments. Cho and Bae [4] proposed a system which drew upon statistical models of flame color combination and image processing techniques. Xu and He [5], using neural-fuzzy grids in residential buildings, developed a fire detection system. This system is designed as alarm system in high-rise building. The main application in high-rise building was reason to detect the fire events. Chen and Chiou [6] also used image processing method for fire detection. But the main advantage of this system is prediction of fire events. Firstly the proposed system by tracking smokes in ambient, predicts the probability of fire event and so is called early fire-detection system. Phillips et al. [7] suggested an algorithm based on color and data obtained from flame movement in video images. Video sequences are analyzed by using image and video processing techniques in their work. Haralick et al. [8] detected fire flames using gray-level matrix (GLCM) and wavelet analysis as well as extraction of contextual features of image. Another method suggested by Healey et al. [9] drew on colored data of image for fire detection. But their work was real-time system that uses image processing techniques and features extracted from fire flames in images. Liu and Ahuja [10], used formal combination of fire for the sake of fire detection in Fourier transform space. They attempt to develop the accuracy and process speed for early fire detection.

In some of these systems, fire detection takes more than tens of seconds. Obviously, this problem would be very serious because the fire detection in early stages is very important in extinguishing fire. Low accuracy in detecting fires is another problem observed in some other methods. For example, the location of growing areas of fire is not addressed in some methods. Thus, there seems to be a need for a smart efficient system which is capable detecting fire accurately and rapidly. Using enhanced image processing techniques, this paper introduces an accurate reliable system.

2. The Proposed System

The method proposed for detection of the fire flames in this paper draw mainly on video processing techniques to detect the target area.

2-1Threshold

The brightness intensity of a gray-scale image can only vary between [0-255]. This is a very useful feature. Partial separation of an image through selecting appropriate threshold is based on dividing the image into background and foreground classes [11]. One of the effective methods in image processing, which is often used to separate the background from image, is threshold method which is based on selecting an appropriate threshold from image histogram. The more careful is this selection, the more accurate will be the accuracy of separating image background from its main framework. In image processing, mainly symmetrical threshold method is used. Image histogram will be split in half at t_0. Then, as to brightness intensity of $t < t_0$, $t > t_0$ average value will be calculated and we will be able to witness the greater average (either on left or right side). Thus, by changing the value of t_0 within certain brightness intensity, the means become equal, or at least, the difference between these two means becomes smaller than a minimal number. However, if partial separation of image is conducted according to brightness intensity, the threshold value or boundary is considered as the basic brightness intensity in division. That is, brightness intensities greater than threshold value will be equal to 1 and brightness intensities less than that will be equal to 0. Thus, we will have a binary image consisting of zero and one elements. Hence, the threshold of the image f (x, y) means to convert it into a binary image according to Eq. (1):

$$g(x,y) = \begin{cases} 1 & f(x,y) \geq T \\ 0 & f(x,y) < T \end{cases} \quad (1)$$

where T is the selected threshold in separating background and foreground of the image. Figure 1 displays the stages of separation process based on overall separation for selecting appropriate threshold.

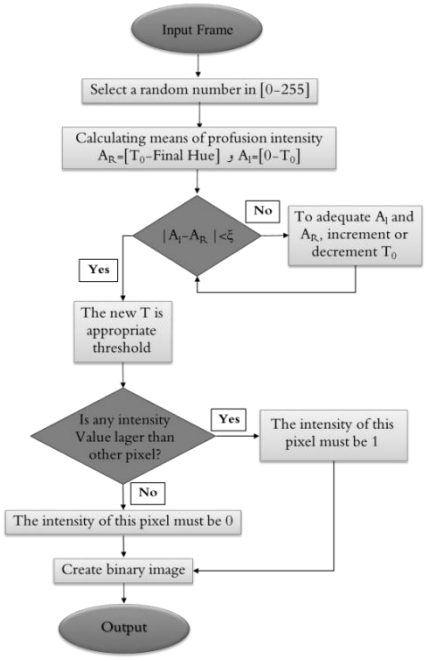


Figure 1. separation based on overall threshold selection

Drawing on this procedure, which is commonly called overall thresholding, the ideal condition takes place when there are sufficient changes between different parts of image. To select proper threshold for separating fire from other parts of image, in some methods, the formal and colored combination property will be used [12]. In Figure 2, the histogram of three constituting spectrum of RGB image for a random frame has been displayed. The main advantage of the histogram is capability detection the adequate threshold in image.

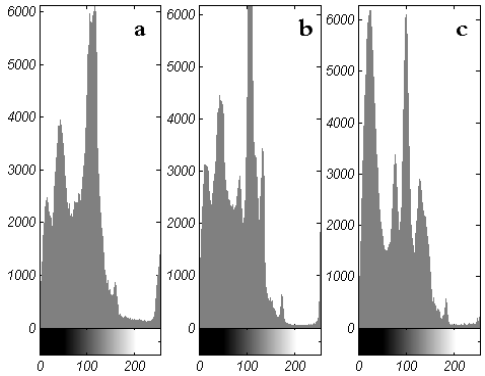


Figure 2. (a) Histogram of Red (b) Green and (c) Blue components in original frame

2-2 Gaussian Mixture Model

The subtraction of the backgrounds of a video sequence from each other is a method by which the target foreground section in each frame is separated. Among the conventional methods of thresholding, Gaussian mixture model is one of the techniques used for separation and display of different background images [13]. According to Eq. (2), Gaussian model for each pixel is the Gaussian density, i.e.:

$$p(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad (2)$$

Where μ is the mean intensity of pixels brightness and σ is the variance. By constructing probability density function (PDF) with μ and σ variables, each pixel in each frame is distributed by K mixture in Gaussian model and the probability of the pixel having the value of X_n at the time of N is calculated according to Eq. (3):

$$P(x_N)=\sum_{j=1}^{K} w_j \eta\left(x_N,\theta_j\right) \quad (3)$$

In which w_K is the weight parameter of K Gaussian factor and $\eta(x,\theta_K)$ is the normal distribution of K factor, which is calculated according to Eq. (4).

$$\eta(x,\theta_k)=\eta(x,\mu_k,\Sigma_k)\frac{1}{(2\pi)^{D/2}\left|\Sigma_k\right|^{1/2}}\exp(Z) \quad (4)$$

μ_k and Σ_k are respectively mean and covariance of K factor and $Z=\left[-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right]$. The number of distribution K is estimated according to the w_K divided σ_k sufficiency function and the first distribution of (B) is used as a foreground model. B function is calculated according to Eq. (5)

$$B=\arg\min_b\left(\sum_{j=1}^{b} w_j > T\right) \quad (5)$$

where T threshold is the lowest decimal value in the foreground model.

2-3 HSV Space

The space which is useful for segmentation of the fire is the conversion of HSV space. Using the features of color space change in HSV, the fire position can be detected by calculating the final area relative to the area of the original image. Using features of color space change in HSV space, the complexity between the level of image and undesired light intensity, which can produces error, is largely reduced. Section H is formed after converting RGB input image into HSV space according to M=max (R, G, B), m=min (R, G, B) and d=M-m. Finally RGB image in form of Figure 1 is converted into HSV space. The values of r, g and b is calculated as a set of $b=(M-B)/d$, $g=(M-G)/d$ and $r=(M-R)/d$. All three H, S and V components change in [1-0] interval.

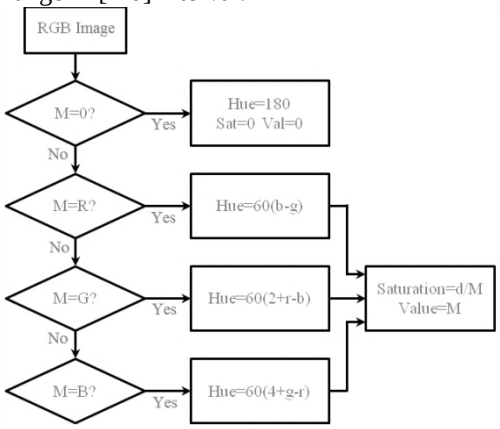


Figure 3. The final diagram of converting RGB space into HSV space for input frames contain fire flames

After converting image from RGB to HSV space, considering the difference between fire color and changes in the image histogram, we will separate the fire. During the conversion, the foreground and background are separated. Figure 4 shows a set of frames taken from a video sequence with threshold implementation as well as application of conversion to HSV space and Gaussian mixture model.

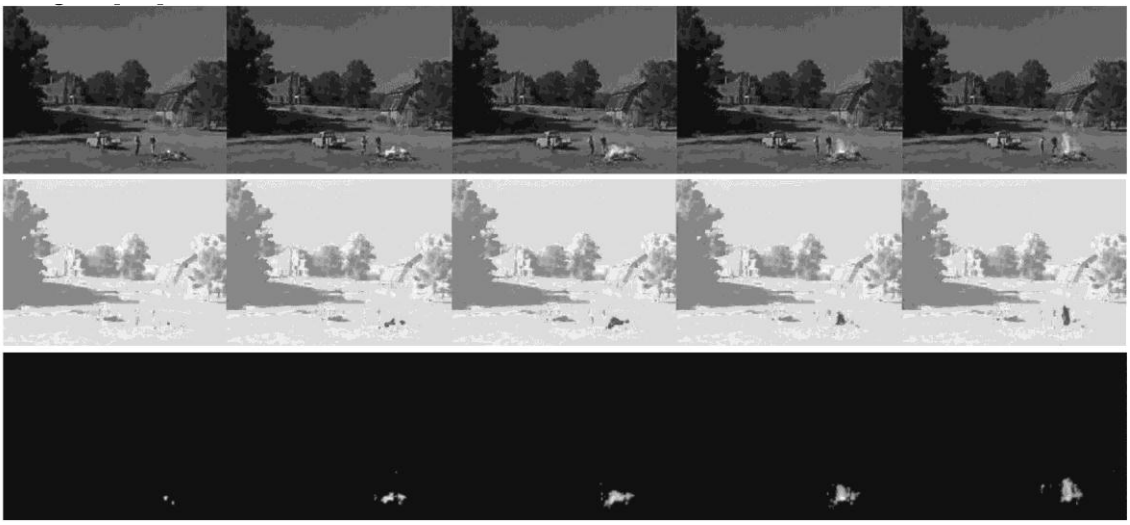


Figure 4. Images from top left to the right show 89 to 94 frames of a sample video sequence from the a fire event, in amidst column frames is converted to the HSV space and in the bottom column, the images produced after using Gaussian mixture model and thresholding has been presented in RGB format.

2-4 Edge Detection and Image Combining

Edge detection is one of the most widely used techniques in image processing in which basic components of the image like the framework of original image are extracted. Edge detection is the use of gradient matrix to detect those image pixels whose brightness intensities have sudden changes in contrast to their adjacent pixels [14]. This technique usually draws on first order and second order derivatives for each pixel. The operator of the first order derivative displays pixels in which the range of brightness intensity is greater than defined threshold value in the same part of image, and second order derivative seeks to find intersecting points of zero-crossing. If we display original image with f (x, y) the first order derivative will be $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$. Pixel gradient of (x, y) is $\nabla \vec{f}$ which has been shown in Eq. (6):

$$\nabla \vec{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (6)$$

Where G_x and G_y respectively are the gradient along the x and y-axis, and the value of gradient operator has been introduced in Eq. (7):

$$\nabla f = \left[G_x^2 + G_y^2\right]^{1/2} \quad (7)$$

In discrete space, variables x and y are replaced by quantized values of discrete m and n that represent the position of pixels in the image. Thus, the gradient is calculated in form of equation (8):

$$\nabla f = \sqrt{f_x^2(m,n) + f_y^2(m,n)} \quad (8)$$

Where f_x and f_y are resulted gradients along X and Y axis respectively. To remove the extra elements that are associated with the threshold process, we use a range between [1-0] in edge detection. That is, if after processing of an edge pixel, the number of its adjacent pixels is less than a basic value, the brightness intensity of the target pixel becomes zero; otherwise, it will retain its previous brightness intensity. These two operators are applied to the images obtained from threshold stage.

For graphical display of fire and precise detection of its location, the brightness intensity of new edge color spectrum is turned into a red spectrum. If the original image is F and restructured colored edge matrix is t, then according to equation (9) we will have:

$$F(t) = 0 \quad (9)$$

Thus, all elements of F matrix which are similar to zero elements of t remain and other elements become zero. Filling polygons of empty and enclosed areas is an algorithm which is carried out according to data structures, morphological properties and partial image processing. Filling the internal area of an enclosed region takes place within two classes of polygon filling and pixel filling [15]. Drawing on morphological reconstruction algorithm [16][17] of image which is based on technique of edge

formal repairing, and using constructive lines of polygon, we will retrieve the data lost during threshold stage. It, thus, increase the accuracy of this stage. In Figure 5, the Filling polygons edge detection and combining images to trace the fire flames is shown.

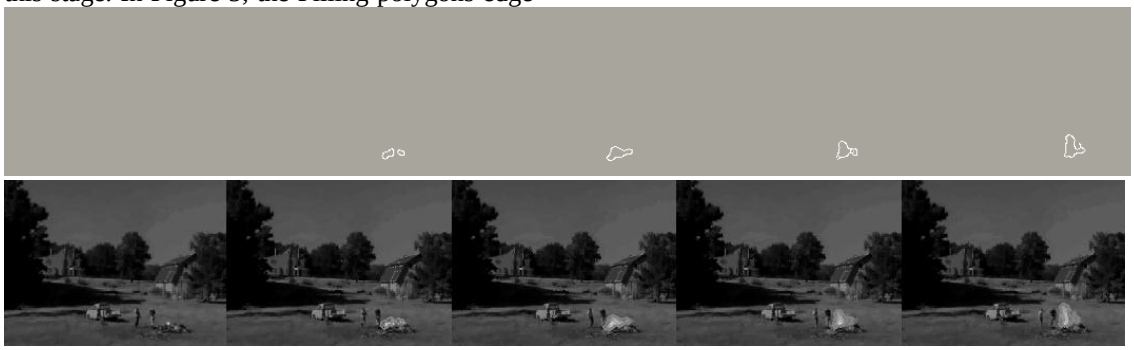


Figure 5. Images from top left to the right show new edge image after filling polygon and and in the bottom column the final composition of the fire location and original image is shown.

3. Performance Analysis

The system was implemented on 57425 video frames taken from AMC channel [18] and video sequences containing the occurrence of the fires. There were 7 fire event cases in video sequences. All sequences were randomly converted into 4 categories of Movie with these details: AVI format, 120 × 160 pixels resolution and 15 fps. In Table 1, mean accuracy (MAC) and detection rate (DR) and false alarm rate (FAR) have been calculated according to Eq. (10) to Eq. (12) which are used for measuring the accuracy of detection in video sequences.

$$AAC = \left(\frac{N_{TP} + N_{TN}}{N_{TP} + N_{TN} + N_{FP} + N_{FN}}\right) \times 100 \quad (10)$$

$$DR = \frac{Number\ of\ True\ Positive}{Number\ of\ Fire\ Video\ Frames} \quad (11)$$

$$FAR = \frac{\sum Detected\ ROI}{Number\ of\ Video\ Frames} \quad (12)$$

In these equations,

$\mathbf{N_{TP}}$ is the number of frames in which fire flames has been detected by the algorithm.

$\mathbf{N_{FN}}$ is the number of frames in which the algorithm has failed to detect fire flames.

$\mathbf{N_{TN}}$ is the number of frames in which there is no sign of fire flames and the algorithm has not detected any by mistake.

$\mathbf{N_{FP}}$ is the number of frames in which there is no sign of fire but the algorithm has detected some by mistake.

And ROI is also the objective area in the image which includes all three signs of fire in the ambient.

Table 1: How average accuracy, detection rate and false alarm rate are calculated.

Video Clips	***No. Frames***	The number frames with fire flames		The number frames without any fire events		Average Accuracy (AAC)	Detection Rate (DR)	False Alarm Rate (FAR)
		N_{TP}	N_{FN}	N_{TN}	N_{FP}			
Movie 1	10755	128	7	283	14	95.13%	94.81%	4.71%
Movie 2	5690	45	0	134	8	95.72%	100%	5.63%
Movie 3	9661	96	3	168	16	93.28%	96.96%	8.69%
Movie 4	8917	89	2	197	9	96.29%	97.80%	4.36%
Total	**35023**	**358**	**12**	**782**	**47**	**95.105%**	**97.39%**	**5.84%**

$SP = \left(\frac{N_{TN}}{N_{TN} + N_{FP}}\right)' 100$ and

$SE = \left(\frac{N_{TP}}{N_{TP} + N_{FN}}\right)' 100$ are specificity and sensitivity which were respectively equal to 94.33% and 96.75%. It should be noted that this method is a new technique and a comparison has been drawn between the performance of this system and other methods used for detection of the fire events. Different methods use various databases. The proposed method in this paper, however, has been applied to a greater database. At first, it should be noted that the use of a single camera would decrease the sensitivity and specificity of the third technique in contrast to other two techniques. Figure 6 displays the implementation of this method as well as

detection of the location and condition of fire spread in different spaces. The experiments show that the algorithm is reliable and accurate. Also the fire experiments show that the shape and color of flame change randomly in fire image sequences.

Figure 6. Implementation the proposed algorithm on 2 clips which contain fire events. The algorithm is implemented on two images in bottom coloumn

3-1 Discussion

The occurrence of error is only natural. Using multiple cameras, which simultaneously identify fire events, increases the accuracy, sensitivity and specificity of this system more than the other two techniques. Table 2 shows the greater ability of the system in detecting the occurrence of a fire. Inconsistency is another problem that may occur when decreasing the frames and the resulted time delay might result in lower sensitivity. In Figure 7 the proposed algorithm has been compared in 15 images using DFBIR and GM [19][20] methods. Randomly selecting 15 images out of 83 tested images, the average accuracy of 95% was compared in Table 2 with valid techniques. The highest accuracy belonged to the jungle image with 515×972 resolution and 99.58% accuracy. The least accuracy in detection of fire belonged to the image of house flames with 689×1034 resolution and 94.07% accuracy.

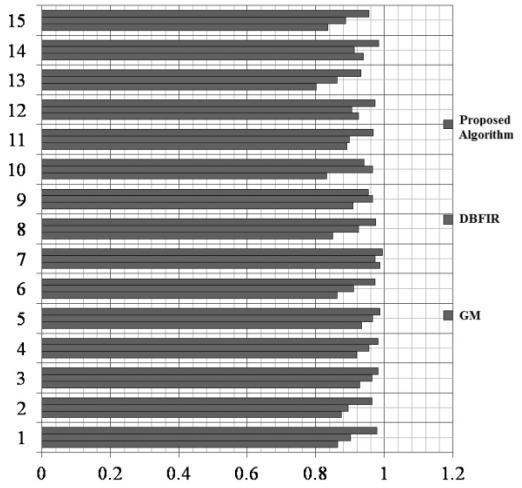


Figure 7. Comparison of the proposed algorithm with DFBIR and GM

Table 2. Comparison of detection rates and false alarm rates in different techniques with proposed algorithm

Technique	Detection Rate	False Alarm Rate
Chen et al [6]	93.90%	66.42%
Celik et al [21]	78.50%	28.21%
Celik et al [22]	97.00%	78.39%
YCbCr [2]	99.00%	31.00%
Lee [23]	85.2%	1.7%
Proposed Algorithm	97.39%	5.84%

4. Conclusions

This paper proposed a system with high accuracy and efficiency which was able to detect fire and its expansion pattern in a short time. In contrast to other fire detection, the performance of the proposed algorithm was at an acceptable level with an average error of 5% and sensitivity of 96%. The implementation of this system in residential and industrial areas as well as open areas such as jungles and farms obviates the need for employment of expensive sensory networks since such factors as low design cost, fast processing and response as well as high accuracy distinguishes this system from similar ones.

5. References

[1] Toreyin, U. and Dedeoglu, Y., "Contour Based Smoke Detection In Video Using Wavelets", IEEE Trans. on Signal Processing, Vol. 42, 2002, pp. 194–196.

[2] Çelik, T. and ÖzkaramanlÆ, H., "Fire Pixel Classification Using Fuzzy Logic And Statical Color Model", ICASSP 2007, Proc IEEE, Conf. on Image Processing, 2007, pp. 1205-1207.

[3] Wirth, M. and Zaremba, R., "Flame region detection based on histogram back projection", IEEE Canadian Conference Computer and Robot Vision, China, pp. 167-174.

[4] Cho, B.H., Bae, J. W., "Image Processing-based Fire Detection System using Statistic Color Model", International Conference on Advanced Language Processing and Web Information Technology, July, 2008, Dalian Liaoning, China pp. 245-250.

[5] Xu, L.M. and He, W., "Application of Fuzzy Neural Network to fire alarm system of high-rise building", Journal of Communication and Computer, Vol.2, No.9, Sep 2005, pp. 18–21.

[6] Chen, T., Wu, P., Chiou, Y., "An early fire-detection method based on image processing" In: Proc. IEEE International Conf. on Image Processing, ICIP 04, 2004, pp. 1707–1710.

[7] Phillips, W., Shah, M., Lobo, N., "Flame recognition in video", Pattern Recognition Lett., Vol. 23, No. (1–3), 2002, pp. 319–327.

[8] Haralick, R.M., Shanmugam, K. and Dinstein, I., "Textural Features for Image Classification", IEEE Transactions on Systems, Man, and Cybernetics, 1973, pp. 610-621.

[9] Healey, G., Slater, D., Lin, T., Drda, B. and Goedeke, A. D., "A system for real-time fire detection", in CVPR '93, 1993, pp. 17-15.

[10] Liu, C. B. and Ahuja, N., "Vision based fire detection", in ICPR '04, 2004, vol. 4.

[11] Anjos, A., Leite, R., Cancela, M. L., Shahbazkia, H., "MAQ–A Bioinformatics Tool for Automatic Macro array Analysis" International Journal of Computer Applications. 2010. Number 7 - Article 1.

[12] Collins, R. T., Lipton, A. J. and Kanade, T., "A system for video surveillance and monitoring", in 8th Int. Topical Meeting on Robotics and Remote Systems, 1999, American Nuclear Society.

[13] Stauffer, C., and Grimson, W.E.L, "Adaptive background mixture models for real-time tracking", Proc. CVPR, Fort Collins, Colorado, USA, 1999, pp. 246–252.

[14] Anderson, W.F., Pfeiffer R.M., Dores G.M., Sherman M.E., "Comparison of age distribution patterns for different histopathologic types of

breast carcinoma", Cancer Epidermal Biomarkers Prev. 2006; Vol. 15 No. 10, 1899-1905.

[15] Pavlidis, T., "Filling algorithms for raster graphics", Computer Graphics and Image Processing, Vol. 10, 1979, pp.126-141.

[16] Soille, P., "Morphological Image Analysis: Principles and Applications", Springer-Verlag, 1999, pp. 173-174.

[17] Vincent, L., "Morphological Gray scale Reconstruction in Image Analysis: Applications and Efficient Algorithms", IEEE Transactions on Image Processing, Vol. 2, No. 2, April, 1993, pp. 176-201.

[18]AMC Channel and Company, available at: http://bobrelease.blogspot.com.

[19] Jun, C., Yang, D., Dong, W., "An Early Fire Image Detection and Identification Algorithm Based on DFBIR Model", World Congress on Computer Science and Information Engineering, 2009, pp. 229-232.

[20] Tingting, L., Shuhai, Q., "An Application of Directional Fractal Parameter Vehicle License Plate Location", Journal of WUT (Information & Management engineering, Vol. 29, No. 3, 2007, pp.14-17.

[21] Celik, T., Demirel, H., Ozkaramanli, H., Uyguroglu, M., "Fire Detection in Video Sequences Using Statistical Color Model", Proc. International Conferences on Acoustics, Speech, and Signal Processing, May 2006, Vol. 2, No. 2, pp. II-213 - II-216.

[22] Celik, T., Demirel, H., Ozkaramanli, H., "Automatic Fire Detection in Video Sequences", European Signal Processing Conference, EUSIPCO-06, Sept. 2006.

[23] Lee Chen, Y., Chin, T. L., "Spatio-Temporal Analysis in Smoke Detection", 2009 IEEE International Conference on Signal and Image Processing Applications, 2009, pp. 80-83.