ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 4, No. 2, 2009, *pp*. 153-160

The Computation of State Variable Model in Software Test Process Using Alternating Group Explicit Iterative Algorithm

Praveen R Srivastava [1, +], Navnit Jha [2] and G Raghurama [3]

[1] Computer science and Information Systems Group, Birla Institute of Technology & Science Pilani-333 031, India

[2] Mathematics Group, Birla Institute of Technology & Science Pilani-333 031, India

[3] Electrical and Electronics Group, Birla Institute of Technology & Science Pilani-333 031, India

(*Received November 23, 2008, accepted February 4, 2009*)

Abstract. The aim of this paper is to construct a parallel algorithm for the model developed in software testing process. The model treats software testing as a control problem. The software under test serves as a controlled object that is modeled as second order equation. The ordinary simulation to such type of model deteriorates in the vicinity of quality, efforts and complexity. The new approach based on parallel alternating group explicit algorithm shows superiority over corresponding sequential algorithm. Computational results are provided to illustrate the viability of the proposed technique.

Keywords: Software testing, State variable, Feedback control, Software process, AGE method, RMSE.

1. Introduction

In this piece of work, we consider the modeling and control of software testing process. Software testing is classified into different category such as structural, functional or random. The internal description of software to generate test cases categories as structural software testing. Many software testing methods are available including software reliability engineering testing, transaction flow testing, equivalence class partition based testing, control flow testing etc. A testing strategy determines what test case should be selected and when software testing should be stopped. The software testing process can be characterized as a feedback system through control theory in order to regulate the testing process. We present a test model based on error reduction as the time increases. Earlier Cangussu *et. al*. [1, 2] has discussed similar model. However, the model fails to explain the behavior of system when number of software defects are inversely related with time as is frequent case. The model numerically leads to a tridiagonal system. The problem of actually determining the solution of such tridiagonal system may be time consuming, particularly when, as is frequently the case, the number of equations is large. Since we need to solve large system of equations, the iterative algorithms are frequently used. The proper choice of the iterative algorithm to be used has a great influence on the amount of computational effort required to solve a given problem. With slowly converging iterative algorithm, the amount of time required may be so large, even with a very fast computer, as to make the solution of problem impractical. Hence the resort to parallel algorithmic approach is imperative.

2. Development of Test Process Model

The software testing process involves development of second order model and is appropriate to capture the essential nature of testing phase. The model is used to estimate the fault in software. During the testing process errors are found and removed. Following parameters and variables are defined as

$r(t)$: Number of software defect remains at time t

s_c : Software complexity

ω_f : Work force

[+] Corresponding author. *E-mail address: praveenrsrivastava@gmail.com*

γ : Constant characterizing overall quality of testing methods

e_{et}: Effective test effort

e_r : Error reduction resistance

e_n : The net effort applied

Using these variables and parameters, we have certain assumptions:

The magnitudes of the rate at which the remaining errors are decreasing is proportional to the net applied effort and are inversely proportional to the time.

$$r'' = \frac{e_n}{t} \tag{2.1}$$

The magnitude of the rate at which the remaining errors are decreasing is inversely proportional software complexity (see Cangussu [4]).

$$r'' = \frac{1}{s_c} \tag{2.2}$$

The magnitude of the effective test effort is proportional to the product of applied work force and the number of remaining errors ζ.

$$e_{et} = r\zeta w_f \tag{2.3}$$

The error reduction resistance is proportional to the error reduction velocity and inversely proportional to the quality of software testing ξ.

$$e_r = \frac{\xi}{\gamma} r' \tag{2.4}$$

Using equations (2.1) - (2.4) and carrying necessary algebra leads to the following model

$$r'' = \frac{\xi}{t\gamma s_c} r' + \frac{\zeta\omega_f}{ts_c} r \tag{2.5}$$

If F_d denotes the disturbance during software testing. Then

$$r'' = \frac{\xi}{t\gamma s_c} r' + \frac{\zeta\omega_f}{ts_c} r + \frac{F_d}{s_c}, \quad t \geq 0 \tag{2.6}$$

The equation (2.6) is written as

$$r'' = d(t)r' + e(t)r + g(t) = \psi(r', r, t) \tag{2.7}$$

where $d(t) = \frac{\xi}{t\gamma s_c}$, $e(t) = \frac{\zeta\omega_f}{ts_c}$, $g(t) = \frac{F_d}{s_c}$

The solution to equation (2.7), usually deteriorates in case of low software quality ($\gamma \to 0$) and/or software complexity (s_c). Difficulties were experienced in the past for the solution of (2.7) in the vicinity of singularity. We overcome these situations by modifying our method based on Taylor's expansion (see Ref. [6]).

Let us define

$$\bar{r}'_k = (r_{k+1} - r_{k-1})/(2h), \; \bar{r}'_{k\pm1} = (\pm 3r_{k\pm1} \mp 4r_k \pm r_{k\mp1})/(2h)$$

$$\bar{G}_{k\pm1} = \psi\left(t_{k\pm1}, r_{k\pm1}, \bar{r}'_{k\pm1}\right), \quad \bar{\bar{r}}'_k = \bar{r}'_k - \frac{h}{20}\left(\bar{G}_{k+1} - \bar{G}_{k-1}\right), \; \bar{\bar{G}}_k = \psi\left(t_k, r_k, \bar{\bar{r}}'_k\right)$$

Then the fourth order finite difference replacement of (2.7) is

$$r_{k-1} - 2r_k + r_{k+1} = \frac{h^2}{12}\left[\bar{G}_{k+1} + \bar{G}_{k-1} + 10\bar{\bar{G}}_k\right], \; k = 1(1)N \tag{2.8}$$

where r_k denote the function satisfying the difference equation at the grids $t_k = kh, k = 0(1)N+1$. The difference scheme (2.8) is of fourth order accurate for the numerical treatment of (2.7). We need the following approximations in order to establish the singular free recurrence relation.

$$d_{k\pm1} = d_k \pm h\, d_k' + \frac{h^2}{2} d_k'' \pm O\left(h^3\right) \tag{2.9a}$$

$$e_{k\pm1} = e_k \pm h e_k' + \frac{h^2}{2} e_k'' \pm O\left(h^3\right) \tag{2.9b}$$

$$g_{k\pm1} = g_k \pm h g_k' + \frac{h^2}{2} g_k'' \pm O\left(h^3\right) \tag{2.9c}$$

Substituting the approximations (2.9) into equation (2.8) and neglecting the higher order terms, we get a linear difference equation of the form

$$a_k r_{k-1} + 2b_k r_k + c_k r_{k+1} = RH_k \ , \ \ k = 1(1)N \tag{2.10}$$

$$a_k = -1 - \frac{h}{24}\begin{bmatrix} 12d_k - 2h\left(2d_k' - d_k^2\right) \\ + h^2(d_k'' - d_k d_k') \end{bmatrix} + \frac{h^2}{12}\left[e_k - \frac{h}{2}\left(2e_k' - d_k e_k\right)\right], \qquad b_k = 1 - \frac{h^2}{3}\left(2d_k' - d_k^2 - 5e_k\right) + \frac{h^4}{12}\left(e_k'' - d_k e_k'\right)$$

$$c_k = -1 + \frac{h}{24}\begin{bmatrix} 12d_k + 2h\left(2d_k' - d_k^2\right) \\ + h^2(d_k'' - d_k d_k') \end{bmatrix} + \frac{h^2}{12}\left[e_k + \frac{h}{2}\left(2e_k' - d_k e_k\right)\right], \qquad RH_k = \frac{-h^2}{12}\left[12g_k + h^2\left(g_k'' - d_k g_k'\right)\right].$$

3. The Alternating Group Explicit Method

The main concern to this section is to present an efficient algorithm based on parallel approach to compute the 3-term recurrence (2.10) developed for Test process model. The linear difference equations (2.10) can be in general expressed as

$$a_k r_{k-1} + 2b_k r_k + c_k r_{k+1} = RH_k \ , \ k = 1(1)N \tag{3.1}$$

The difference equation (3.1) in matrix notation is

$$\boldsymbol{Ar} = \boldsymbol{R} \tag{3.2}$$

where $\boldsymbol{A} = \begin{bmatrix} 2b_1 & c_1 & & & \\ a_2 & 2b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{N-1} & 2b_{N-1} & c_{N-1} \\ & & & a_N & 2b_N \end{bmatrix}$, $\boldsymbol{r} = \left[r_1, r_2, \cdots, r_N\right]^{\mathrm{T}}$,

$\boldsymbol{R} = \left[RH_1 - a_1 r_0, ,.., RH_N - c_N r_{N+1}\right]^{\mathrm{T}}$

We split the matrix $\boldsymbol{A}$ as follows

$$\boldsymbol{A} = \boldsymbol{G}_1 + \boldsymbol{G}_2 \tag{3.3}$$

Where

$$\boldsymbol{G}_1 = \begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & & & \bigcirc & \\ & & b_3 & c_3 & & \\ & & a_4 & b_4 & & \\ & & & & \ddots & \\ & \bigcirc & & & b_{N-1} & c_{N-1} \\ & & & & a_N & b_N \end{bmatrix}$$

$$\boldsymbol{G}_2 = \begin{bmatrix} b_1 & & & & & \\ & b_2 & c_2 & & \bigcirc & \\ & a_3 & b_3 & & & \\ & & & \ddots & & \\ & & & b_{N-2} & c_{N-2} & \\ & \bigcirc & & a_{N-1} & b_{N-1} & \\ & & & & & b_N \end{bmatrix}$$

Assuming $\boldsymbol{G}_1 + \omega\boldsymbol{I}$ and $\boldsymbol{G}_2 + \omega\boldsymbol{I}$ are non singular matrices, then parallel method based on alternating group explicit algorithm is given by

$$(\boldsymbol{G}_1 + \omega\boldsymbol{I})r^{(k+1/2)} = \boldsymbol{R} - (\boldsymbol{G}_2 - \omega\boldsymbol{I})r^{(k)} \tag{3.4a}$$

$$(\boldsymbol{G}_2 + \omega\boldsymbol{I})r^{(k+1)} = \boldsymbol{R} - (\boldsymbol{G}_1 - \omega\boldsymbol{I})r^{(k+1/2)}, \quad k = 0, 1, 2, \ldots\ldots. \tag{3.4b}$$

where $\omega > 0$ is an acceleration parameters and $r^{(k+1/2)}$ is an intermediate vector(Evans [6])). Since $\boldsymbol{G}_1 + \omega\boldsymbol{I}$ and $\boldsymbol{G}_2 + \omega\boldsymbol{I}$ are non singular, the equations (3.4a) and (3.4b) in the explicit form

$$r^{(k+1/2)} = (\boldsymbol{G}_1 + \omega\boldsymbol{I})^{-1}\left[\boldsymbol{R} - (\boldsymbol{G}_2 - \omega\boldsymbol{I})r^{(k)}\right] \tag{3.5a}$$

$$r^{(k+1)} = (\boldsymbol{G}_2 + \omega\boldsymbol{I})^{-1}\left[\boldsymbol{R} - (\boldsymbol{G}_1 - \omega\boldsymbol{I})r^{(k+1/2)}\right] \tag{3.5b}$$

Simplifying equations (3.5a) and (3.5b), we obtain following algorithm

First sweep: For $j = 1(2)N-1$

Let

$$\Pi = (b_j + \omega)(b_{j+1} - \omega) - c_j a_{j+1} \neq 0$$

$$A = R_j - \beta a_j r_{j-1}^{(k)} - (b_j - \omega) r_j^{(k)}$$

$$B = R_{j+1} - (b_{j+1} - \omega) r_{j+1}^{(k)} - \delta c_{j+1} r_{j+2}^{(k)}$$

where $\beta = \begin{cases} 0 & \text{if } j = 1 \\ 1 & \text{otherwise} \end{cases}$

$$\delta = \begin{cases} 0 & \text{if } j = N-1 \\ 1 & \text{otherwise} \end{cases}$$

Then

$$r_j^{(k+1/2)} = \left(A(b_{j+1} + \omega) - B c_j\right)/\Pi$$

$$r_{j+1}^{(k+1/2)} = \left(B(b_j + \omega) - A a_{j+1}\right)/\Pi$$

Second sweep: For $j = 1$

$$r_1^{(k+1)} = \frac{\left(R_1 - (b_1 - \omega) r_1^{(k+1/2)} - c_1 r_2^{(k+1/2)}\right)}{(b_1 + \omega)}$$

For $j = 2(2)N - 2$

Let

$$\Pi = (b_j + \omega)(b_{j+1} - \omega) - c_j a_{j+1} \neq 0$$

$$A = R_j - a_j r_{j-1}^{(k+1/2)} - (b_j - \omega) r_j^{(k+1/2)}$$

$$B = R_{j+1} - (b_{j+1} - \omega) r_{j+1}^{(k+1/2)} - c_{j+1} r_{j+2}^{(k+1/2)}$$

Then

$$r_j^{(k+1)} = \left(A(b_{j+1} + \omega) - B c_j\right) / \Pi$$

$$r_{j+1}^{(k+1)} = \left(B(b_j + \omega) - A a_{j+1}\right) / \Pi$$

For $j = N$

$$r_N^{(k+1)} = \frac{\left(R_N - (b_N - \omega) r_N^{(k+1/2)} - a_N r_{N-1}^{(k+1/2)}\right)}{(b_N + \omega)}$$

4. Computational illustrations

To prove the efficiency of the implementation of the parallel alternating group explicit algorithm, all parameters with different valid values are taken. All results of numerical experiments, which were gained from implementation of the algorithms has been recorder in the tables. In all cases, initial guess is taken as zero vector and the iterations were stopped when difference of two consecutive iterative values is less than 10^{-10} was achieved. The acceleration parameter $\omega = 0.9$ is chosen for parallel AGE algorithm. The computational results shows superiority over corresponding sequential algorithm (Kai-Yuan Cai, *et. al.* [5]). The exact solution $r(t) = \exp(t^4)$ is chosen. The right hand side function and boundary conditions may be obtained using the exact solution as a test procedure. The root mean square errors (RMSE), minimum number of time required to compute in terms of iterations I_P for parallel case and I_S for sequential case are presented in Table 1, 2 and 3.

5. Conclusion

From Table 1, it is evident that as software quality increases the average number of iterations in parallel case I_P decreases more as compared to corresponding sequential iterations I_S. Table 2 shows that as work force increases the number of iterations in parallel and sequential case decreases and consequently execution time diminish with increase in work force. Table 3 confirms the superiority of parallel iterations on sequential one. Consequently high software complexity leads to very low execution time in parallel case as compared to its counterpart.

Table 1

N	I_S	I_P	$RMSE$
$\gamma = 0.5$			
20	49	210	3.722e-05
30	100	214	7.796e-06
40	171	221	2.586e-06
50	261	232	1.103e-06
60	369	248	5.508e-07
$\gamma = 0.6$			
20	60	158	2.772e-05
30	125	163	5.891e-06
40	214	173	1.976e-06
50	326	192	8.502e-07
60	461	236	4.277e-07
$\gamma = 0.7$			
20	72	125	2.156e-05
30	150	132	4.648e-06
40	256	148	1.576e-06
50	390	193	6.834e-07
60	551	246	3.454e-07
$\gamma = 0.8$			
20	83	103	1.737e-05
30	174	113	3.796e-06
40	297	148	1.299e-06
50	452	205	5.667e-07
60	638	301	2.879e-07

$s_c = 2, w_f = 4, \xi = 20, \zeta = 4$

Table 2

N	I_S	I_P	$RMSE$
$w_f = 2$			
20	62	158	2.475e-05
30	129	164	5.517e-06
40	220	174	1.915e-06
50	335	195	8.444e-07
60	474	242	4.325e-07
$w_f = 3$			
20	61	158	2.607e-05
30	127	164	5.656e-06
40	217	174	1.927e-06
50	331	193	8.391e-07
60	467	239	4.258e-07
$w_f = 4$			
20	60	158	2.772e-05
30	125	163	5.891e-06
40	214	173	1.976e-06
50	326	192	8.502e-07
60	461	236	4.277e-07
$w_f = 5$			
20	59	157	2.956e-05
30	123	163	6.193e-06
40	211	172	2.052e-06
50	321	191	8.742e-07
60	454	233	4.363e-07

$s_c = 2, \gamma = 0.6, \xi = 20, \zeta = 4$

Table 3

N	I_S	I_P	$RMSE$
$s_c = 2$			
20	60	158	2.772e-05
30	125	163	5.891e-06
40	214	173	1.976e-06
50	326	192	8.502e-07
60	461	236	4.277e-07
$s_c = 4$			
20	132	68	9.512e-06
30	278	128	2.367e-06
40	475	224	8.668e-07
50	722	342	3.940e-07
60	1018	482	2.057e-07
$s_c = 8$			
20	252	115	6.117e-06
30	529	247	1.477e-06
40	902	423	5.306e-07
50	1366	643	2.381e-07
60	1921	904	1.234e-07
$s_c = 16$			
20	405	186	9.621e-06
30	850	396	2.023e-06
40	1447	677	6.591e-07
50	2189	1026	2.739e-07
60	3074	1443	1.322e-07

$w_f = 4, \gamma = 0.6, \xi = 20, \zeta = 4$

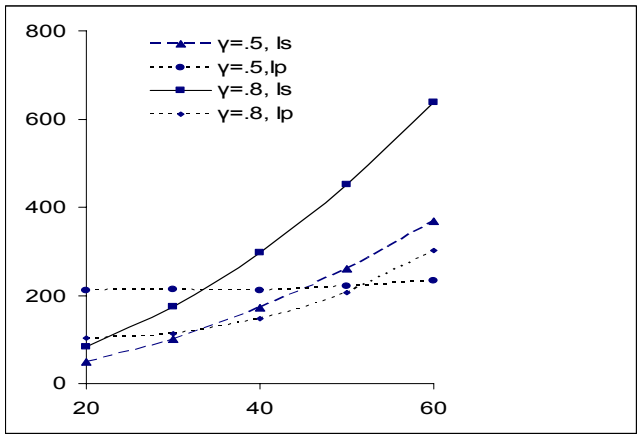

Fig. 1

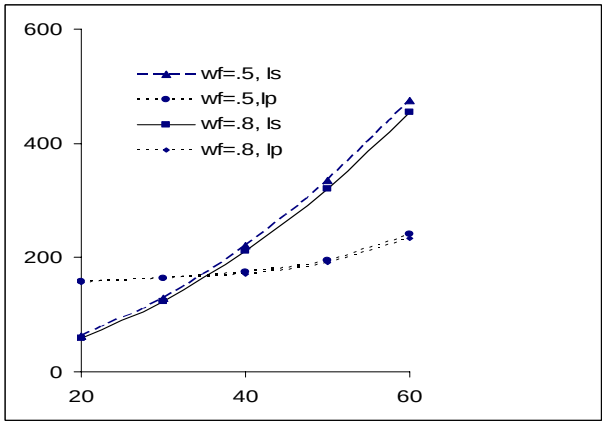

Fig. 2

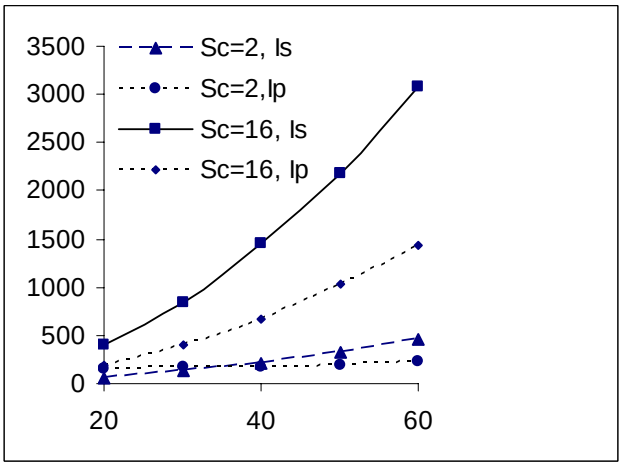

Fig. 3

1. References:

[1] Joao W. Cangussu, Raymond A. DeCarlo and A. P. Mathur. A State Variable Model for the Software Test Process. *Proc. of 13th International Conference on Software and System Engineering and their applications (ICSSEA).* Paris-France, December 2000.

[2] Joao W. Cangussu. A Formal Model for the Software Test Process. *IEEE Transactions on Software Engineering.* 2002, **28**(8): 782-796.

[3] Kai-Yuan Cai. Optimal software testing and adaptive software testing in context of software cybernetics. *Information and Software Technology*. 2002, **44**: 841-855.

[4] Joao W. Cangussu. Modeling and Controlling the software test process. *Proc. 23rd International Conference on Software Engineering(ICSE'01)*, 2001.

[5] Kai-Yuan Cai, T Y Chen, Yong-Chao Li, Wei-Yi Ning and Y T Yu. Adaptive testing of software components. *ACM Symposium on Applied Computing*, 2005.

[6] Evans D. J.. Group explicit method for solving large linear system. *Int. J. Comput. Math.* 1985, **17**: 81-108.

[7] Kai-Yuan Cai, Yong-Chao Li, and Ke Liu. Optimal and adaptive testing for software reliability assessment. *Information and Software Technology*. 2004, **46**: 989-1000.