

Using an Path-Comparison Tracing Algorithm to Solve the most K Critical Paths Problem of Stochastic Networks

Weng-Ming Chu, Chun-Min Hsu, and Hwi-En Tseng

Abstract—The focus of this study is how to determine the most K Critical Paths (KCP) for stochastic network and we propose a Path Comparison Tracing Algorithms (PCTA) to solve this problem. In past studies, the well known Critical Path Method (CPM) has been the general method used for determining the critical path in deterministic networks but it is not proper to be used in stochastic network. Meanwhile, instead of solving single critical path in stochastic network, the KCP would be more practical for decision makers to face the real world cases. PCTA is based on the Label-Correcting Tracing Algorithm (LCTA) and modified Stochastic Dominated Operation (SDO) in our algorithms. Our numerical results were compared with results of the PERT and the Monte Carlo Simulation (MCS) of 20,000 samples and we concluded that our proposed approach is efficient for solving the KCP problem of stochastic networks.

Keywords—K Shortest Path, Critical Path, Stochastic network, Stochastic Domination, PERT.

I. INTRODUCTION

In general, all the activities' durations are random variables in stochastic network, the completion time of completion path should has its own probability distribution function (*pdf*). Therefore, all completion paths could have their probability to be the critical path in the stochastic network. Hence, it is useless for project managers to determine single critical path by using the method of CPM. In this paper, we focus on finding the most K Critical Paths (KCP) in stochastic networks instead of single critical path, it is mainly based on the reasons of Eppstein (1994,1999) which are as follows:

- 1) Keep more flexibility: The completion time of the critical path may possibly be influenced by various factors that project managers impossibly think of all. Therefore, we think it is better to keep flexibility by selecting KCP instead of one critical path in the network.
- 2) Sensitivity analysis: As the project managers intent to query the influences by some specific factors, it is helpful to clearly realize the paths' changing by picking the KCP.

Weng-Ming Chu was with the *Department of Electronic Commerce Hsing Kuo University of Management, Tainan, Taiwan of R.O.C.* (corresponding author to provide phone: 886-6-2873613; e-mail: chuwendming@gmail.com).

Chun-Min Hsu and Hwi-En Tseng were with the Department of Industrial Engineering and Management, Chin-Yi University of Technology, Taichung, Taiwan of R.O.C. (e-mail: lamo7272@yahoo.com.tw and hwaien@ncut.edu.tw, respectively).

Besides, if the *pdf* s of KCP are obtained, the decision maker can get more detailed information of network management.

The path completion time in stochastic network is a random variable which can not be directly calculated by the summing all activities' durations on the critical path as in a deterministic network. According to Michel, Francesco & Salvatore (2000), they have denoted the calculation of the path completion time of stochastic network is the NP hard problem. This calculation burden would grow exponentially with the increasing of activity numbers.

For most project managers, the PERT technique is the most general method to be applied for estimating of the project completion time in networks. This technique applies many assumptions to estimate the probability distribution of the completion time. The most two of them are listed as follows:

- 1) All the activities and paths in network are independent each other.
- 2) The completion time of each path is normally distributed and its mean and variance are calculated as the sum of the activity mean and variance of duration respectively in the path.

The calculations of above are based on the Central Limit Theory (CLT) where the number of activity should be required large enough.

With above assumptions, it is not practical in the real world. For the first assumption, the network paths are usually not independent with each other since they could share a couple of the same activities in the network. For the second assumption, it should require the big size networks. Beyond that, the CLT is not necessary since, there exists the effect of the Longest Path Bias (LPB) or Shortest Path Bias (SPB) incurred from Max or Min operation respectively in the network, This phenomenon has been found and noted by the earlier Fisher Tippet Theorem (1982), Klingel (1966), David's (1981) Order Statistics Theorem, Dodin & Sirvanci (1990), Mehrotra et al. (1996), and Pontrandolfo (2000). Klingel(1966) suggested to apply Monte Carlo Simulation (MCS) to solve LPB or SPB problem. Dodin & Sirvanci (1990) used dominate path concept and extreme value theory to estimate the mean and the variance of the completion time in stochastic network. Pontrandolfo(2000) applied the PERT-path developed by Giovanni(1994) to solve the shortest path problems in stochastic network. However, all

the techniques mentioned above are too complicate and heavily loading in calculations.

Hoffman & Pavley(1959) are the earliest one to solve the most K Shortest Paths (KSP) problem in deterministic network. The following literatures have developed other different algorithms: Martin(1984) applies the path-deletion algorithm, Hadjiconstantinou & Christofides(1999) use the method of Katoh, Ibaraki & Mine(1982) to obtain the KSP in the networks, Rink, Rodin & Sundarapandian(1998) proposed a algorithm called Double-Sweep to deal with KSP, and, after that, Francesca, et al.(2001) applied the Label-Correcting and Label-Setting methods to solve KSP problem. He compared the results of using different methods and found the pure threshold method with the best performance. Dodin(1984) employs Stochastic Dominance Operation (SDO) to determine the ordering-of-duration among paths. But it requires the *pdf* of all activities' durations must be symmetrically distributed where the transitivity could be applied to decide the ordering of different paths. Tan, Cheng & Gao(2001) use the network expand method and Stochastic Dominance operation to solve the KSP of transportation in stochastic model.

Since the criteria and characteristics of approximation of the Longest Path Problem (LPP) are quite similar with the Shortest Path Problem (SPP). Therefore, we abbreviate these two problems as the terminology of "Critical Path Problem (CPP)". That is the reason why we use the KCP instead of KSP.

In this study, we propose an Path Comparison Tracing Algorithms (PCTA) to solve the KCP. PCTA is based on the Label-Correcting Tracing Algorithm (LCTA) developed by Yao et al. (2007), we modify it by adding the Stochastic Dominated Operation (SDO) on Max or Min operation among various incident paths in the network. Different from Dodin(1984), we have further applied Pairwise Comparing Matrix (PCM) in SDO to get rid of problem of transitivity on the ordering-of-variable. Before the executions of KCP, we have to 'discretize' the continuous *pdf* of each activity duration into discrete model first, since KCP has also to approximate the completion time of stochastic network as LCTA did. The more detail will be described in the following sections.

In this research, the network was in the form of a two-terminal Activity-On-Node(AON) and represented as $G=(V, E)$, with V and E as the set of nodes and the edge in the network, respectively. Each node be designed an integer index number i and represented as $node(i)$, the start node and the terminated node are presented as $node(1)$ and $node(N)$, respectively. We summarize all the necessary notations as below:

X_{ji} : The activity duration of two neighboring nodes (j, i).

Y_j^i : The j^{th} incidence path duration of $node(i)$.

γ_i : The accumulated output duration of $node(i)$.

B_i : The precedence nodes set of $node(i)$.

A_i : The successor nodes set of $node(i)$.

The rest of paper is organized as follows: In section II, we give a detailed description of the problem happened with KCP.

Section III introduces method of our modified SDO for variables comparison. The section IV presents the full algorithm analysis of the PCTA for solving the KCP in stochastic network. Section V, we demonstrate our proposed KCP algorithm via numerical experiments and also compare its performances with results out of PERT and MCS of 20,000 samples. Finally, we address our concluding remarks in Section VI.

II. THE PROBLEM OF FINDING KCP IN STOCHASTIC NETWORK

The KCP problem of stochastic network is not as easy as selecting the most K longest or shortest path duration from all paths in the deterministic network, the selection must be done through the complex comparison among the plentiful network paths and their duration are all be specified as probability distribution function (*pdf*). As we well know, the of approximation of *pdf* will be effected by the Longest Path Bias (LPB) or Shortest Path Bias (SPB) incurred from Max or Min operation in the network, furthermore, there exist path dependence problems in the stochastic network, it also bias the approximation of path duration (Yao et al., 2007, Yao & Chu, 2007). In addition, the sorting of path duration is not as easy as directly comparing the mean value of duration variables (Dodin, 1984). In this section, we will describe the above problem in a simple example.

A. The LPB and SPB

The earlier Fisher Tippet Theorem (1928), Klingel (1966), Order Statistics Theorem (1981), Dodin & Sirvanci (1990), Mehrotra et al. (1996) and Pontrandolfo (2000) have noticed this problem. In this research, we name the bias generated between the PERT and Max operator or Min operator as the Longest Path Bias (LPB) and the Shortest Path Bias (SPB), respectively. The PERT model is the general method used for CPP. It directly summarizes all node duration of a completion path to get its duration, then, compares it with other paths and chooses the longest or shortest one as the critical path. This simple method may cause great error in the stochastic network, since it did not been considered the existence of the Longest Path Bias (LPB) and Shortest Path Bias (SPB) generated from the Max operator and Min operator in the network. By taking Fig. 1 as an example, we will show you how does the LPB influences the completion time approximating in the networks.

The path <1-2-3-4> and <1-3-4> have encountered at $node(3)$, if we assume the incident paths $Y_2^3 > X_{13}$, the longest path would be path <1-2-3-4> and the completion time is calculated as $Y_2^3 \oplus X_{34}$ in the viewpoint of the PERT. But if we commit Max operator on $node(3)$ and get $\gamma_3 = \text{Max}(X_{13}, Y_2^3)$ where $\gamma_3 > Y_2^3$ in mathematical realization, the real completion time is calculated as $\gamma_3 \oplus Y_2^3$ which is greater than the result of PERT. In the same way, the real shortest path <1-3-4> would be smaller than the result of PERT while we apply the Min operator at $node(3)$.

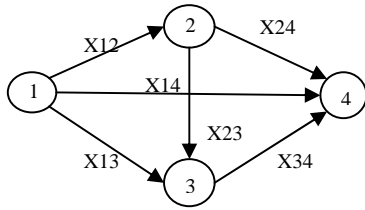


Fig. 1 A small example network with four activities

From above, it clearly indicates that the LPB should be considered while dealing with KCP problem. We can also note that the value of the LPB is something related to overlap area of *pdf* of two stochastic variables, it increases by when these two variables have close means and large variances. On the other hand, the LPB can be neglected when these two variables are almost “separate” from each other, with significantly different means and small variances.

B. Path Dependence Problems of Stochastic Network

The most of the literatures did not take into account the dependency of paths in the stochastic network. In this subsection, we have experiment to show you that it could also cause *significant* bias in the approximated of the path duration.

By referring to Fig. 2, we assume all the activity duration are exponent distribution, their mean value are shown in the figure. Provided that two paths <1,2,3,4,5,7> and <1,2,3,4,6,7> are two independent paths where,

$$\gamma_4 = X_{12} \oplus X_{23} \oplus X_{34}, \quad \gamma_5 = \gamma_4 \oplus X_{45}, \quad \gamma_6 = \gamma_4 \oplus X_{46},$$

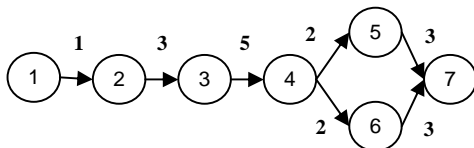
$$Y_1^7 = \gamma_5 \oplus X_{57}, \quad Y_2^7 = \gamma_6 \oplus X_{67}, \quad \text{and} \quad \gamma_7 = \max\{Y_1^7, Y_2^7\}.$$


Fig. 2 An AoA network with seven activities

After the calculations of above, we solved the $\gamma_7 = 17.69$, which is different from the result $\gamma_7 = 15.89$ generated from the Monte Carlo Simulation (MCS) of 20,000 samples (A deviation of 11.33%). We name such an error as the Shared Activity Bias (SAB) incurred out of the extra LPB or SPB of the shared activities in a stochastic network., since the LPB or SPB is very sensitive to the variance of the operant while executing the Max or Min operator. According to Fig. 3, path<1-2-3-4-5-7> and path<1-2-3-4-6-7> have shared the sub-path <1-2-3-4>, the variance of the sub-path <1-2-3-4> has counted twice due to the independent consideration. Therefore, the extra LPB or SPB will be generated and it is the bias generated form.

C. The Problem of Order Decision on Path Duration

We have to determine the most K critical paths from the stochastic network for solving the KCP problem. In general and

instinctive thinking, it can be done by just only selecting the most K longest duration of completion paths in the network. But it is not as easy as what above saying due to the LPB and path dependent conditions. In this subsection, we have another network example (Fig. 3) to explain it.

Let all the activities’ durations as exponential distribution where their mean of duration are shown on the arc in Fig. 3. If we apply PERT model to determine the most longest path, it would be the <1,2,6> and the mean of completion time is 9.1. In the other way, if we think over LPB effect, the Max operation will be executed at *node(5)* and *node(6)*, and we get the output result of *node(6)* is 14.64 which significantly different from result of PERT model. Furthermore, the output of *node(5)* is calculated as 7.2 where we sense that the combination of path <1, 3, 5, 6> and <1, 4, 5, 6> are longer than the path <1, 2, 6>, it also contradicts the result of PERT model.

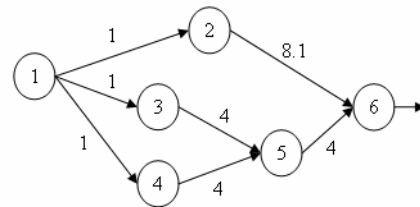


Fig. 3 A 6-nodes network

How come it could generate the quite a different results in a simple network of just only 6 nodes? It is because of the LPB generated from Max operation. If we applied the Monte Carlo Simulation (MCS) with 20000 samples to simulate, we find the path <1, 2, 6> with Path Critical Index (PCI) 0.39, and path <1, 3, 5, 6> and <1, 4, 5, 6> that have the same PCI value 0.305. Meanwhile, the mean completion time for *node(6)* is 14.82, which is close to the result done by Max operator. Fig. 4 is illustrated the *pdf* comparison of completion time done by PERT, Max and MCS operators.

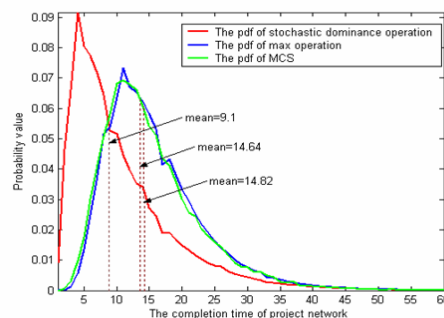


Fig. 4 The *pdf* comparing among Stochastic Dominance, Max and MCS operators

From the experiment above, obviously, the network completion time is not dominated by the path <1,2,6> any more, even though it has got with the highest PCI value. It means that we can’t directly select path <1,2,6> as the most critical path of the network though it does in this case. The LPB generated from the Max operation of path <1, 3, 5, 6> and <1,

4, 5, 6> could influence the amount of domination on the network completion time for path <1,2,6>. Therefore, the order of paths seems should be counted by the path domination on the network completion time. In the last section, we have developed a algorithm to approximate the scaling of domination on the network completion time for each path.

III. THE MODIFIED STOCHASTIC DOMINANCE OPERATION

The Stochastic Dominance Operator (SDO) is used to decide the ordering among stochastic variables, Dodin(1984) and Tan, Cheng & Gao(2001) have applied it to solve the most K critical paths of the stochastic networks. Let Y_j^i represents the duration of j^{th} incidence path of $node(i)$, and we have two incidence paths Y_1^j and Y_2^j . If Y_1^j dominates Y_2^j , then we could say that Y_1^j has a greater possibility than Y_2^j to be the longest path in the network. We represent it as $Y_1^j \succ Y_2^j$ and the Stochastic Dominance Probability Value (SDPV) represented as follows:

$$\Pr(Y_1^j \succ Y_2^j) \geq \Pr(Y_2^j \succ Y_1^j),$$

$$SDPV_1^j = \Pr(Y_1^j \succ Y_2^j), \quad SDPV_2^j = \Pr(Y_2^j \succ Y_1^j) = 1 - SDPV_1^j \quad (1)$$

where $\Pr(\cdot)$ is denoted as the probability value of the internal function.

Oppositely, if Y_1^j dominates Y_2^j in accordance with the shortest path, we could say that Y_1^j has a greater possibility than Y_2^j to be the shortest path. It is represented as $Y_1^j \prec Y_2^j$ and the SDPV is expressed as:

$$\Pr(Y_1^j \prec Y_2^j) \geq \Pr(Y_2^j \prec Y_1^j)$$

$$SDPV_1^j = \Pr(Y_1^j \prec Y_2^j), \quad SDPV_2^j = \Pr(Y_2^j \prec Y_1^j) = 1 - SDPV_1^j \quad (2)$$

The SDO can only be applied for SDPV between two variables, since it could happen with the *Transitivity* problem in case that the variable number is greater than 2. Provided that we have $Y_1^j \succ Y_2^j$ and $Y_2^j \succ Y_3^j$, does it refer to $Y_1^j \succ Y_3^j$? According to Dodin (1984), the main condition of *Transitivity* requires all the variables to be symmetric in their probability distribution functions. To the best of our knowledge, it is impossible for all paths' duration of stochastic network to fulfill this requirement.

In order to solve above *Transitivity* problem of SDO, we construct a $n \times n$ Pairwise Comparison Matrix (PCM) for n stochastic variables $Z = (z_1, z_2, \dots, z_n)$, and we let $W = (w_1, w_2, \dots, w_n)$ to be the ideal ratio scale among Z , where $a_{ij} = w_i / w_j$, then the ordering between Z could be judged by W . With above definition, the PCM can be represented as P :

$$P = \begin{bmatrix} 1 & a_{12} & \dots & a_{1n} \\ a_{21} & 1 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & 1 \end{bmatrix} = \begin{bmatrix} w_1/w_1 & w_1/w_2 & \dots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \dots & w_2/w_n \\ \vdots & \vdots & \ddots & \vdots \\ w_n/w_1 & w_n/w_2 & \dots & w_n/w_n \end{bmatrix} \quad (3)$$

If all variables fulfill the characteristic of *Transitivity*, we would have each pair variables as follow:

$$a_{ij} \cdot a_{jk} = (w_i / w_j) \cdot (w_j / w_k) = (w_i / w_k) = a_{ik} \quad (4)$$

In the case of above, the matrix P could be named as *Consistence Matrix* and the ideal ratio scale W can be directly calculated from elements a_{ij} of P . Or we use another method, we let $W = (w_1, w_2, \dots, w_n)$, then the deductive equations are as follow:

$$\sum_{j=1}^n a_{ij} w_j \frac{1}{w_j} = n, \quad i = 1, \dots, n$$

$$\therefore a_{ij} \cdot \frac{w_j}{w_i} = 1, \quad \therefore \sum_{j=1}^n a_{ij} w_j = n w_i, \quad i = 1, \dots, n, \quad (5)$$

$$PW = nW, \quad W = (w_1, \dots, w_n)^T$$

The eigenvalue of P is n and (w_1, w_2, \dots, w_n) is solved as its correspondent eigenvector.

According to equation (4), if any one of elements of PCM does not fulfill the operation of *Transitivity*, it means that there is little different between a_{ij} and ideal w_i / w_j . Then, the eigenvalue of P would have little deviation from n which denoted as n' and, meanwhile, the correspondent eigenvector of n' would also be adjusted itself to be $W' = (w'_1, w'_2, \dots, w'_n)$, but it still can be used as the approximation of ratio scale of (z_1, z_2, \dots, z_n) .

The above method of PCM can be applied in this study to get rid of the *Transitivity* problem of SDO and to get the ratio scales (w_1, w_2, \dots, w_n) of n stochastic variables. Each element a_{ij} of PCM represents the SDPV which is generated from the SDO between variable z_i and z_j , where $i, j \in [1, n]$. But the PCM generated out of SDO can't directly take into above calculation since it is different from one of equation (3), it is listed as follow:

$$P' = \begin{bmatrix} 1 & a_{12} & \dots & a_{1n} \\ (1-a_{12}) & 1 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ (1-a_{1n}) & (1-a_{2n}) & \dots & 1 \end{bmatrix} \neq \begin{bmatrix} w_1/w_1 & w_1/w_2 & \dots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \dots & w_2/w_n \\ \vdots & \vdots & \ddots & \vdots \\ w_n/w_1 & w_n/w_2 & \dots & w_n/w_n \end{bmatrix}$$

$$\forall a_{ij} \in [0, 1] \quad (6)$$

The elements of P' need be transferred to follow the form of w_i/w_j by simply replacing each element value of a_{ij} as follows:

$$w_i/w_j, \text{ where } w_i = a_{ij}, w_j = a_{ji} = (1 - a_{ij}) \quad (7)$$

Therefore, P' can't be transformed to P'' which follows the form of equation (5) and can be solved the eigenvalue and the correspondent eigenvector. P'' is represented as follow:

$$P'' = \begin{bmatrix} 1 & a_{12}/a_{21} & \dots & a_{1n}/a_{n1} \\ a_{21}/a_{12} & 1 & \dots & a_{2n}/a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}/a_{1n} & a_{n2}/a_{2n} & & 1 \end{bmatrix} = \begin{bmatrix} 1 & w_1/w_2 & \dots & w_1/w_n \\ w_2/w_1 & 1 & \dots & w_2/w_n \\ \vdots & \vdots & \ddots & \vdots \\ w_n/w_1 & w_n/w_2 & & 1 \end{bmatrix} \quad (8)$$

We have further explanation through Fig. 5 where SDO was executed at $node(i)$ for totally 4 incidence paths $Y_1^i, Y_2^i, Y_3^i, Y_4^i$.

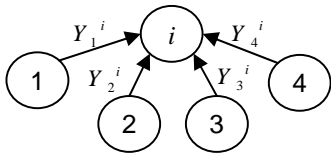


Fig. 5 4 incident paths of $node(i)$

At first, each pair of two different incidence paths executes SDO individually, and we have:

$$\begin{aligned} a_{12} &= \Pr(Y_1^i > Y_2^i) = 0.2, & a_{21} &= 1 - a_{12} = 0.8 \\ a_{13} &= \Pr(Y_1^i > Y_3^i) = 0.15, & a_{31} &= 1 - a_{13} = 0.85 \\ a_{14} &= \Pr(Y_1^i > Y_4^i) = 0.123, & a_{41} &= 1 - a_{14} = 0.877 \\ a_{23} &= \Pr(Y_2^i > Y_3^i) = 0.248, & a_{32} &= 1 - a_{23} = 0.752 \\ a_{24} &= \Pr(Y_2^i > Y_4^i) = 0.2, & a_{42} &= 1 - a_{24} = 0.8 \\ a_{34} &= \Pr(Y_3^i > Y_4^i) = 0.333, & a_{43} &= 1 - a_{34} = 0.667 \end{aligned}$$

The modified 4×4 PCM was formed as:

$$P'' = \begin{bmatrix} 1 & 0.25 & 0.17 & 0.14 \\ 4 & 1 & 0.33 & 0.25 \\ 6 & 3 & 1 & 0.5 \\ 7 & 4 & 2 & 1 \end{bmatrix} \quad (10)$$

According equation (5), the eigenvalue of P is calculated as $n' = 4.102$ where n' is approach to $n = 4$, it means there is little deviation between a_{ij} and ideal w_i/w_j in the P . The adjusted ratio scale is calculated as the eigenvector $W' = (0.062, 0.097,$

$0.224, 0.617)$ which is the ratio scale of incidence paths $Y_1^i, Y_2^i, Y_3^i, Y_4^i$. Obviously, the order sequence of these paths is listed as $(Y_4^i, Y_3^i, Y_2^i, Y_1^i)$ from large to small.

It has noted that the KCP could be solved by selecting the most K paths from network which are gotten with the highest *Domination* on network completion time. The *Domination* can be quantified as value of SDPV calculated from SDO on the nodes of more than two precedent nodes in the network. For conveniently representation of the modified SDO in network analysis, let $node(i)$ has m incident paths $Y^i = \{Y_q^i \mid \forall q \in B_i\}$, we defined $W' = Dominate(i)$ for the following algorithm of this research.

IV. PATH COMPARISON TRACING ALGORITHM (PCTA)

When the networks become complex and large, the above method for KSP would be not easy at all. It should need an efficient algorithm to visit each node in the network. Therefore, we apply LCTA (developed by Yao et al., 2007) and modified it by adding with modified SDO of section III into the algorithm. LCTA should has to transfer the network into Expanded-Tress Structure (ETS), setup a specified data structure for each node and systematically visit each node through upward and downward tracing procedures. We will give them a detailed description in this section.

A. Establish the Expanded-Tree Tracing Structure

LCTA should require the network be transferred into a tree structure by referring to the successive and precedent relations among nodes, we name it as the Expanded-Tree Structure (ETS). Each node of the network should be designated a unique node number index. Node 1 and node N are represented as the start node and terminal node, respectively. Fig. 6 is an ETS example transformed from the network of Fig. 1.

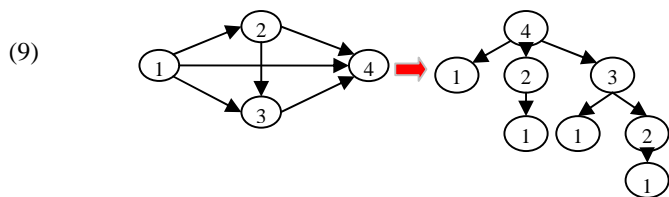


Fig. 6 An example of ETS transformation

B. The Data Structure of the Network Nodes

There are a lot of calculations and generated information need to be recorded during the LCTA operation. Therefore, we propose a node data structure for each node to fulfill these requirements. For fulfilling the requirements of solving KSP, the contents of the data structure are expressed as:

- 1) i : index number of $node(i)$.
- 2) B_i : set of precedent nodes of $node(i)$.
- 3) X_{ji} : the activity duration between nodes j and i .
- 4) $Y^i = \{Y_j^i \mid j \in B_i\}$: the incident path duration set of node

- i.
 - 5) $node(i).flag$: visited flag set of incident paths of $node(i)$ where all are initially set as 0. When the j^{th} incident paths of $node(i)$ have been visited, the relative j^{th} position of $node(i).flag$ will be set as 1.
 - 6) $finish_flag_i$: visited flag value of $node(i)$ initially set as 0. When $node(i)$ has been visited, it will be set as 1.
 - 7) γ_i : the output duration of node i .
 - 8) $SDPV^i = \{SDPV_j^i \mid j \in B_i\}$: the set of SDPV of node(i) .
($SDPV^i = Dominate(i)$)
 - 9) $node(i).path$: set of all sub-paths from $node(0)$ to $node(i)$. Each sub-path is recorded as a set of node index numbers in the sequence of $node(0)$ to $node(i)$.
- $node(i).TSDPV$: set of temporary SDPV of sub-paths in $node(i).path$.

C. The two Tracing Procedures

We have two tracing procedures for the LCTA which can sequentially visit each node of the network. They can systematically and efficiently reach each node of the network. During the visiting of $node(i)$, the realization of completion time will be recorded in γ_i , $SDPV^i = Dominate(i)$ will be executed, the sub-paths of $node(i).path$ will be updated with $node(i)$ and $node(i).TSDPV$ will be accumulated with $SDPV^i$. Finally LCTA will reach the terminal node N and final SDPV of all the completion paths can be solved.

Each node has a unique index label. The label will be put into a *queue* for recording the path-track whenever the tracing procedures pass through it. The queue labels will form a path sequence, and this is taken as the history path-track so that the tracing procedures can find their way back to the original node where they started. The tracing procedures initially starts from the terminal node of the network, then go downward until they reach the start node 1. The *queue* is constructed by a *stack structure* which actually is a serial memory, where the data accessing follows the rule of the Last In/First Out (LIFO). The label recording and reading are denoted as *push_stack()* and *pull_stack()*, respectively. We have a control pointer to indicate the LCTA current position of the *queue* accessing, and it can go forward or backward follow the requirement of the tracing procedures. By the time the LCTA reaches the starting node, the tracing begins to bounce upward and starts the node visiting and approximation calculations.

With above queue mechanism, LCTA has further applied Post-Order Tracing Algorithm (POTA) on the ETS so as to let LCTA could efficiently and systematically visit each node in the network. Before visiting the $node(i)$, LCTA has to confirm that all the child nodes of $node(i)$ have been visited, and the child nodes should be visited in sequence from left to right. This is the important rule of POTA. By taking the Fig. 1 as example, the tracing sequence of the nodes is shown in Fig. 7. The tracing starts from terminal $node(4)$ and the visited sequence is in the direction of the blue line which forms the contour of the ETS. LCTA completes this tracing work by the

recursive downward and upward tracing, until all the nodes have been visited. The tracing procedure will finally return back to terminal $node(4)$.

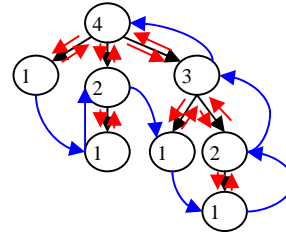


Fig. 7 The tracing sketch map of Fig. 1

Our purpose is to solve the KCP through above tracing procedure. Therefore, LCTA must check two conditions for the current visited node $node(i)$ during the iterative tracing, as follows:

A. Condition 1:

Have all the child nodes $node(j) \in B_i, \forall j$ of $node(i)$ been visited (it can be done by checking the $finish_flag_i$)?

✓ Yes:

- 1) To execute the $\gamma_i = Min(Y^i)$ or $\gamma_i = Max(Y^i)$, which upon whether the scenario is to search for the shortest path or longest path, respectively.
- 2) To execute the $SDPV^i = Dominate(i)$ for all the precedent paths Y^i of $node(i)$.
- 3) $Path_Calculation(node(i).path, node(i).TSDPV)$ for updating the $node(i).path$ and $node(i).TSDPV$, we will give they the detail explanation in the latter subsection.
- 4) Retrieve the index number of the father node of $node(i)$ and let it be the new current node. This is done by $i=pull_stack()$.
- 5) If $stack\ structure \neq \phi$, modify $node(i).flag$ and go back to step A. Otherwise, go to step C.

✓ No:

- 1) To execute the calculation of $Y_j^i = \gamma_j \oplus X_{ji}$ for the specified incidence path from $node(j)$ to $node(i)$.
- 2) If the discrete sample points of Y_j^i over 100, then execute $DRT(Y_j^i)$.
- 3) Record the current $node(i)$ into the *stack structure* by $push_stack(i)$.
- 4) Find the next-to-visit child node of $node(i)$ by referring to $node(i).flag$.

B. Condition 2:

Is the current node to be the start node ($i=1$)?

✓ Yes:

- 1) Let $\gamma_1 = 0$ and $finish_flag_1 = 1$.
- 2) Retrieve the index number of the father node of $node(1)$ and let it be the new current node. This is done by $i=pull_stack()$.

- 3) Go to step A.
 ✓ No:
 Modify $node(i).flag$ and go to step A.

C. Stop the tracing.

We summarize the above process and functions into two procedures: Upward_Tracing (last_node, current_node) and Downward_Tracing (current_node). The current_node is the currently visited node in the tracking process; the last_node is the precedent visited node of the current_node. We show their algorithm as Algorithm 1:

Algorithm 1:

$i = current_node;$

$j = last_node;$

Upward_Tracing (j, i)

while ($finish_flag_N = 1$) **do**

if ($finish_flag_i = 1$)

$$Y_j^i = \begin{bmatrix} \mu(\gamma_j) \\ 1 \end{bmatrix} \oplus X_{ji};$$

¹ $Shared_flag = 0;$

Else

$$Y_j^i = \gamma_j \oplus X_{ji};$$

$Shared_flag = 0;$

End

if (the samples of Y_j^i is over 100?)

² $DRT(Y_j^i);$

End

if ($finish_flag_i = 0$)

$j = i;$

$i = Select_node(i);$ /select the next child node of $node(i)$ by following rule of POTP

$push_stack(i);$

Downward_Tracing (j, i);

Else

$\gamma_i = Min(Y^i)$ or $\gamma_i = Max(Y^i);$

$SDPV^i = Dominate(i);$

$Path_Calculation(node(i).path, node(i).TSDPV);$

$finish_flag_i = 1;$

$Shared_flag = 0;$

$j = i;$

$i = pull_stack();$

End

End

Downward_Tracing (j, i)

while ($finish_flag_N = 1$) **do**

if ($finish_flag_i = 1$)

$Shared_flag = 1;$

$j = i;$

$i = pull_stack();$

Upward_Tracing(j, i);

Else

if ($i = 1?$)

$\gamma_1 = 0;$

$node(1).TSDPV = \{1\};$

$node(1).path = \{<1>\};$

$finish_flag_i = 1;$

$j = i;$

$i = pull_stack();$

Upward_Tracing(j, i);

Else

$push_stack(i);$

$i = j;$

End

End

End

The LCTA starts from the terminal node N and executes the Downward_Tracing procedure. Until the Downward_Tracing procedure reaches the starting node, we have the realization time of $node(1)$ by $\gamma_1 = 0$, $node(1).path = \{<1>\}$ and $node(1).TSDPV = \{1\}$ from our node structure, and then begin the Upward_Tracing procedure. During the Upward_Tracing procedure, if there exists any child nodes of the current node that has not been visited, then the algorithm will transfer to the Downward_Tracing procedure. Through the recursive interaction between the Downward_Tracing and the Upward_Tracing procedures, the LCTA will finally stop at the terminal node. Meanwhile, the visited node i would be checked and executes $Dominate(i)$ for $SDPV^i$, and then, update $node(i).path$ and $node(i).TSDPV$ through $Path_Calculation(node(i).path, node(i).TSDPV)$.

D. The Selection of KCP from PCTA

Form above section, we have $Path_Calculation(node(i).path, node(i).TSDPV)$ to accumulate the paths from $node(1)$ to $node(i)$ and record them into $node(i).path$. Beyond that, the $Path_Calculation$ could also execute the value of SDPV for each paths within $node(i).path$. After PCTA has finished its tracing procedures, all the completion paths will be recorded in terminal node of $node(N).path$ and their correspondent final SDPV would be recorded in $node(N).TSDPV$. The paths of KCP are selected from $node(N).path$ by referring the front K highest SDPV from $node(N).TSDPV$.

During the tracing of PCTA, whenever the child nodes of the current visited $node(i)$ has finished their visiting, the $node(i)$ could be promised to execute $Path_Calculation(node(i).path, node(i).TSDPV)$. According to Fig. 7, PCTA starts its tracing from $node(4)$, when PCTA touches down to $node(1)$, it bounces up and begins

¹The $Shared_flag$ is used for dealing with dependent path problem in LCTA, reader could refer to Yao et. al. (2007) for more detail.

² Whenever the discrete sample data of duration over 100, we resample it to 100 in case of overburden on the computation of the LCTA algorithm. Reader could refer to Yao et. al. (2007) for more detail.

to record the node index number and execute the SDPV. The $node(1)$ is the first one to execute the function of $Path_Calculation$, it records the node index number as $node(1).path=\{<1>\}$ and its correspondent SDPV is as $node(1).TSDPV=\{1\}$. Following the rule of POTP, each tracing paths and its SDPV would be recorded sequentially and saved into the structure of $node(i).path$ and $node(i).TSDPV$, respectively. We have the step-by-step description below:

- 1) PCTA starts its tracing from $node(4)$ and goes downward until it touches $node(1)$.
- 2) $node(1)$ executes $Path_Calculation$ and records $node(1).path=\{<1>\}$, $node(1).TSDPV=\{1\}$, then back to position of $node(4)$.
- 3) Since the incident paths of $node(4)$ have not all been visited yet, the $node(4).path$ and $node(4).TSDPV$ are still empty.
- 4) PCTA goes down to $node(1)$ through $node(2)$ by following the rule of POTP. Since $node(1)$ has been visited, LCTA upwards and back to $node(2)$.
- 5) Since $node(2)$ has only 1 incident path, $node(2)$ executes $Path_Calculation$ and commit $node(2).path=\{<1,2>\}$, $node(2).TSDPV=\{1\}$, then back to position of $node(4)$.
- 6) Since the incident paths of $node(4)$ have not all been visited yet, the $node(4).path$ and $node(4).TSDPV$ are still empty.
- 7) PCTA goes downward to $node(1)$ through $node(3)$, since $node(1)$ has been visited, then upward and back to $node(3)$.
- 8) Since the incident paths of $node(3)$ have not all been visited yet, the $node(3).path$ and $node(3).TSDPV$ are still empty.
- 9) PCTA goes down to $node(2)$, since $node(2)$ has been visited, it goes upward and back to $node(3)$ again.
- 10) $node(3)$ executes $Path_Calculation$ and records two paths in $node(3).path=\{<1,3>, <1,2,3>\}$ by referring to contents of $node(1).path$ and $node(2).path$. The SDO is also executed at $node(3)$ as:

$$\begin{aligned}
 SDPV_1^3 &= \Pr(Y_1^3 \succ Y_2^3) \\
 SDPV_2^3 &= 1 - SDPV_1^3 \\
 SDPV_{<13>} &= SDPV_1^3 \times node(1).TSDPV = SDPV_1^3 \\
 SDPV_{<123>} &= SDPV_2^3 \times node(2).TSDPV = SDPV_2^3 \\
 \text{where } Y_1^3 &= X_{12} \oplus X_{23}, \quad Y_2^3 = X_{13}
 \end{aligned} \tag{11}$$

- 11) After above executions, we have $node(3).TSDPV=\{SDPV_1^3, SDPV_2^3\}$.
- 12) PCTA goes back to $node(4)$ and execute $Path_Calculation$, since all the incident paths have been visited. $node(4)$ records four paths in $node(4).path=\{<1,4>, <1,2,4>, <1,3,4>, <1,2,3,4>\}$ by referring to contents of $node(1).path$, $node(2).path$ and $node(3).path$. The SDO is also executed at $node(4)$ as:

$$\begin{aligned}
 SDPV_1^4 &= \Pr(Y_1^4 \succ Y_2^4 \ \& \ Y_1^4 \succ Y_3^4), \quad Y_1^4 = X_{12} \oplus X_{24} \\
 SDPV_2^4 &= \Pr(Y_2^4 \succ Y_1^4 \ \& \ Y_2^4 \succ Y_3^4), \quad Y_2^4 = X_{14} \\
 SDPV_3^4 &= \Pr(Y_3^4 \succ Y_1^4 \ \& \ Y_3^4 \succ Y_2^4), \quad Y_3^4 = \gamma_3 \oplus X_{34}, \\
 \gamma_3 &= \text{Max}[\gamma_2, X_{13}], \quad \gamma_2 = X_{12} \oplus X_{23}
 \end{aligned} \tag{12}$$

- 13) The correspondent SDPV of paths within $node(4).path$ are calculated as:

$$\begin{aligned}
 SDPV_{<14>} &= SDPV_1^4 \times node(1).TSDPV = SDPV_1^4 \\
 SDPV_{<124>} &= SDPV_2^4 \times node(2).TSDPV = SDPV_2^4 \\
 SDPV_{<134>} &= SDPV_3^4 \times node(3).TSDPV(1) \\
 &= SDPV_3^4 \times SDPV_1^3 \\
 SDPV_{<1234>} &= SDPV_3^4 \times node(3).TSDPV(2) \\
 &= SDPV_3^4 \times SDPV_2^3
 \end{aligned} \tag{13}$$

Due to the *Transitivity* problem of SDO, the SDO of $node(4)$ should be required to form a 3x3 Pairwise Comparison Matrix (PCM) at first, then execute the eigenvector of that matrix by referring to equation (7) to get the values of $SDPV_1^4$, $SDPV_2^4$ and $SDPV_3^4$. After that, since the path $<1, 3, 4>$ and $<1, 2, 3, 4>$ share same incident path X_{34} , they have to dispatch the value of SDPV at $node(3)$. Therefore, their final SDPV should need accumulated calculation with values of $node(3).TSDPV$.

Obviously, the completion path has the larger value of SDPV, it would has the more probability to be the critical path in the network and the highest *Domination* on network completion time. Therefore, the KCP could be solved by selecting the paths of the front K highest SDPV in the terminal node of $node(N).TSDPV$.

V. NUMERICAL EXPERIMENTS

There are 480 instances of activity networks in our experiments which are generated in Demeulemeester and Herroelen (1997) by the software ProGen (developed by Kolish 1996). In this study, those network instances are designed and classified with the factor of network complexity, they are divided into three levels: 1.5, 1.8 and 2.0, the node number are either 32 or 48 in the networks.

We assume that the activity duration of each node is exponentially distributed with mean being randomly determined within the range of [1,10]. The number of preceding and successor nodes for any node in network is also randomly determined within the range of [1,3]. We conducted our experiments by Intel Pentium-4 2.0 GHZ CPU, 512MB and the program by Matlab Ver. 6.5 package.

We take the results of Monte Carlo Simulation (MCS) with 20,000 samples as benchmark, to compare with the results of PCTA, PERT and Dodin Algorithm (1984). The networks have been separated into three parts according to different network complexity level (1.5, 1.8 and 2.0), each level includes 160 network instances. For each instance network, the experiments

are separated into three sections: we pick up the most 3, 6 and 9 completion paths to compare with MCS, respectively. The comparison is to find the match numbers of those the most critical path with MCS's. If the generated K critical paths of instance network completely match with those of MCS, it would be got a hit, and we calculate the totally hit numbers as the performance. The compared results are shown on the Table I.

TABLE I
THE COMPARED RESULTS OF THE KCP

Approaches Network complexity		PERT model	Dodin Algorithm	PCTA
K=3	1.5	126/160	128/160	143/160
	1.8	113/160	114/160	127/160
	2.0	102/160	106/160	116/160
	average	341/480 (71%)	348/480 (72.5%)	386/480 (80.4%)
K=6	1.5	130/160	129/160	149/160
	1.8	115/160	116/160	131/160
	2.0	106/160	108/160	121/160
	average	351/480 (73.1%)	353/480 (73.5%)	401/480 (83.5%)
K=9	1.5	135/160	131/160	155/160
	1.8	120/160	118/160	140/160
	2.0	116/160	109/160	132/160
	average	371/480 (77.3%)	358/480 (74.9%)	427/480 (88.9%)

From Table I, It is reasonable that we find the results of three methods are all getting better with growing number of critical paths. Our proposed PCTA is always the best among these methods which has performance of match hit over 85% in average. It is because that PERT and Dodin Algorithm did not consider the factor of path dependent in their approximation of completion time. Beyond that, we have further added PCTA with the modified SDO operator into approximation, it will lift the precision more or less. Though it is not good enough for that PCTA has not 100% match hit with results of MCS, it still better than the other methods as Dodin's algorithms (73.6%) and PERT (73.8%) in average.

PCTA is also good at the performance of algorithm efficiency, readers can tell them by referring to Yao et. al.(2007) and Yao & Chu (2007) where their used network algorithm are based on LCTA. From Table II, we have further experiments on the comparison of running time and completion time approximation among PCTA, PERT and Dodin's algorithm.

TABLE II
THE RUNNING TIME COMPARISON

Network complexity	Running Time Comparison			Percentage error of mean time of network completion time		
	$\frac{MCS}{PCTA}$	$\frac{MCS}{DA}$	$\frac{MCS}{PERT}$	$\frac{ PCTA-MCS }{MCS}$	$\frac{ DA-MCS }{MCS}$	$\frac{ PERT-MCS }{MCS}$
1.5	288	212	2101	2.6%	22%	20%
1.8	282	230	3122	4.7%	23%	21%
2.0	529	411	6605	7.5%	28%	24%
Average	366	284	3943	4.9%	24.3%	21.7%

From Table II, though the running time of PERT are faster than PCTA, PCTA is still faster than Dodin's algorithm and 366 times faster than MCS. Its average error mean of network completion time between with MCS is less than 4.9% in average, which is very much better than that of Dodin's algorithm (24.3%) and PERT (21.7%).

VI. CONCLUSION

In this study, we have noted that any one of completion path could possibly be the most critical path in stochastic network, therefore, the determination of the most K critical paths (KCP) instead of single one would be more practical for decision makers to face the cases of real world. According to the explanation of Eppstein (1994,1999), KCP could has benefits of flexibility and sensitivity for stochastic network analyses. But, for solving KCP in stochastic network, it would be a hard work, since it has to face with the problems of LBP, SPB and path dependent in the network. Most of literatures of concerning KCP did not think over them all, that is reason why we have developed an efficient algorithm (PCTA) to overcome these problems concurrently. Beyond that, The Stochastic Dominance Operation (SDO) is well know for comparing two stochastic variables, but it can not apply for variables of more than 2, since the Transitivity characteristic did not exist consequentially among variables. For dealing with this problem, we apply Pairwise Comparison Matrix (PCM) to store the SDO results of each pair variables, and compute the eigenvector of PCM as approximated comparison results of those variables. From above descriptions, there are two main contributions in this paper: The first, we have purposed a modified SDO that can commit the comparison among stochastic variables, no matter what types of probability distribution function (*pdf*) they got with. And the second, we have applied the modified SDO and developed an efficient algorithm (PCTA) for solving KCP in stochastic network, no matter how big size the network, it can complete it in reasonable running time. In the experiments, we have applied 480 instances and compared the performances of PCTA, PERT and Dodin(1984) with MCS with 20000 samples, and find that PCTA not only get the highest average match hit of 85% on the 3, 6 and 9 most critical paths but also the running speed is 366 times than the MCS. In conclusion, Comparing with PERT and Dodin Algorithm(1984), our proposed PCTA is more efficient

and accurate for solving the most K former critical paths problem of the stochastic network.

REFERENCES

- [1] David, H. A., "Order statistics", 2d ed., Wiley, 1981.
- [2] Demeulemeester E.L. Herroelen W.S. "New benchmark results for the resource-constrained project scheduling problem", *Management Science*, Vol. 43, n 11, 1485-1492, 1997.
- [3] Dodin B.M., "Determining the most K former critical paths in PERT networks", *Operations Research*, 32, n4, 859-877, 1984.
- [4] Dodin B.M., Sirvanci M., "Stochastic networks and the extreme value distribution", *Computers & Operations Research*, v 17, n 4, 397-409, 1990.
- [5] Eppstein D., "Finding the K shortest paths", *Proceeding 35th IEEE Symposium Foundations of Computer Science*, 154-165, 1994.
- [6] Eppstein D., "Finding the K shortest paths", *SIAM J. Comput*, 28, 2, 652-673, 1999.
- [7] Fisher, R., L. Tippet, "Limiting Forms of the Frequency Distribution of the Largest or Smallest Member of a Sample", *Proceedings of the Cambridge Philosophical Society*, 24, 1, 180-190, 1928.
- [8] Francesca G., Roberto M., Balerio L., Antonio, P., "A class of label-correcting methods for the K shortest paths problem", *Operations Research*, 49, 3, 423-429, 2001.
- [9] Giovanni M., "PERT-path network technique: a new approach to project planning", *International Journal of Project Management*, 12, n2, 89-99, 1994.
- [10] Hadjiconstantinou E., Christofides N., "An Efficient Implementation of an Algorithm for Finding K Shortest Simple Paths". *Networks*, 34, 88-101, 1999.
- [11] Hoffman W., Pavley R., "A method for the solution of the Nth best path problem", *J. Assoc Commun*, March 6, 506-514, 1959.
- [12] Kato N, Ibaraki T., Mine H., "An efficient algorithm for K shortest simple paths", *Networks*, 12, 411-427, 1982.
- [13] Kolisch, R., Sprecher, A., "PSPLIB-A project scheduling problem library", *European Journal of Operational Research*, 96, 205-216, 1996.
- [14] Klingel, A.R., Jr., "Bias in PERT project completion time calculations for a real network", *Management Science*, 13, n4, 476-489, 1966.
- [15] Martin E.Q.V., "An algorithm for ranking paths that may contain cycles", *European Journal of Operational Research*, 18, 123-130, 1984.
- [16] Mehrotra K., Chai J. & Pillutla S., "A study of approximating the moments of job completion time in PERT networks", *Journal of operations Management*, 14, 277-289, 1996.
- [17] Michel C., Francesco L.P., Salvatore T., "A hierarchical approach for bounding the completion time distribution of stochastic task graphs", *Performance Evaluation*, 41, 1-22, 2000.
- [18] Pontrandolfo, P., "Project duration in stochastic networks by the PERT-path technique", *International Journal of Project Management*, 18, 215-222, 2000.
- [19] Rink K.A., Rodin E.Y., Sundarapandian V., "A simplification of the double-sweep algorithm to solve the k-shortest path problem", *Applied Mathematics Letters*, 13, 77-85, 2000.
- [20] Tan G.Z., Chen Z., Cao W., "Distributed and parallel K shortest paths algorithm", *Proceeding of the 6th International Conference for Young Computer Scientist*, 454-458, 2001.
- [21] Yao, M.J., Chu, W.M., Tseng, T.Y., "A Label-Correcting Tracing Algorithm for the Approximation of the Probability Distribution Function of the Project Completion Time", *Journal of Chinese Institute of Industrial Engineers*, 24-2, p.153-165, March 2007.
- [22] Yao, M.J., Chu, W.M., "A New Approximation Algorithm for Obtaining the Probability Distribution Function of the Project Completion Time", *Computers and Mathematics with Applications*, 54-2, p. 282-295, July 2007.

Weng Ming Chu was born in Taiwan, Jul. 1962. He received the Ph.D. degree in the Department of Industrial Engineering and Enterprise Information, Tunghai University, Taiwan. Now, he is interested in project management, supply chain management, inventory control, and telecommunication networks.

Dr. Chu used to be an officer of Airforce for airplane radar training. Now, he has retired and to be an assistant professor at Hsing Kuo University of Management in Tainan.