Performance Utility-Analysis of Multi-State Systems

Shaomin Wu and Ling-Yau Chan

***Abstract*—This paper defines a new utility importance of a state of a component in multi-state systems. This utility importance overcomes some drawbacks of a well-known importance measure suggested by William S. Griffith (*J. Applied Probability,* 1980). The relationship between this new utility importance and the Griffith importance is studied and their difference is illustrated with examples. The contribution of an individual component to the performance utility of a multi-state system is discussed. Examples show that a meaningful index for measuring the performance of individual components in a multi-state system can hardly be defined in general, without considering the actual values of the utility levels and the distributions of the component-states in the system. An example illustrates how genetic algorithm, simulated annealing, and tabu search can be used in selecting components and defining the position order of components so that the performance utility of a multi-state system is optimized.**

***Index Terms*—Multi-state systems, state importance, utility importance.**

ACRONYMS[1]

GA	genetic algorithm
SA	simulated annealing
TS	tabu search.

NOTATION

K	number of nonzero states of the system, $1 \leq K < \infty$
j	an integer, $0 \leq j \leq K$
a_j	system utility level when the system is in state j; $a_K \geq \cdots \geq a_1 \geq a_0 \geq 0$.
N	number of components available for use, $1 \leq N < \infty$
n	number of components in the system, $1 \leq n \leq N$
i	an integer, $1 \leq i \leq n$
M	number of nonzero states of each component, $1 \leq M < \infty$
m	an integer, $0 \leq m \leq M$
$\mathbf{X}_i$	a random variable which represents the state of component i in the system
x_i	a value of $\mathbf{X}_i$, $0 \leq x_i \leq M$
$\mathbf{X}$	$(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$: row vector of the $\mathbf{X}_i$
X	$(x_1, x_2, \ldots, x_n)$: row vector of the x_i
$(m_i, \mathbf{X})$	$\mathbf{X}$: $\mathbf{X}$ when its coordinate #i has the value m
$\phi(\mathbf{X})$	a system structure function; it represents the state of the system as a function of $\mathbf{X}$
$\phi(X)$	value of $\phi(\mathbf{X})$ when $\mathbf{X} = X$
$p_{i,j}$	$\Pr\{\mathbf{X}_i = j\}$
$R_{i,j}$	$\Pr\{\mathbf{X}_i \geq j\}$
U	$\sum_{j=0}^{K} a_j \cdot \Pr\{\phi(\mathbf{X}) = j\}$, performance utility function (see [12]) of a system
$\mathbf{I}^G(i)$	Griffith importance vector of component i
$\mathbf{I}^{\mathrm{UI}}(i)$	utility importance of component i.

Manuscript received September 17, 1999; revised June 9, and September 25, 2001. Responsible Editor: S. Rai.

S. Wu is with the Department of Computer Science, University of Bristol, Bristol BS8 1UB, U.K. (e-mail: Shaomin@cs.bris.ac.uk).

L.-Y. Chan is with the Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Hong Kong (e-mail: pLYChan@hku.hk).

Digital Object Identifier 10.1109/TR.2002.805783

[1]The singular and plural of an acronym are always spelled the same.

I. INTRODUCTION

IMPORTANCE measures are used to measure the effect of the reliability of individual components on the system reliability. From the design view-point, it is crucial to identify the weaknesses of the system and how failure of each individual component affects proper functioning of the system; so that efforts can be spent properly to improve the system reliability [1]. Various authors have introduced importance-measures in binary systems. Birnbaum-importance measures the contribution of component-reliability to the system reliability [1], [2]. Structural-importance measures the topographic importance of a position in the system [3], [4]. Criticality-importance corresponds to the conditional probability of failure of a component, given that the system has failed [1], [2]. Joint-importance measures how components in a system interact and contribute to the system-reliability [5], [6].

A binary system is formed from 2-value logic, e.g., on/off, and functioning/not-functioning. Although such a system has many practical applications, a model based on dichotomizing the system states is often over-simplified and insufficient for describing many commonly encountered situations in real life; as a result, multi-state systems are frequently required. In a multi-state system, the components and/or the system performance have more than 2 states. There are numerous examples of multi-state systems, with more than 2 ordered or unordered states at the system level, the sub-system level, or the component level. A power plant which has states 0, 1, 2, 3, 4 that correspond to generating electricity of 0%, 25%, 50%, 75%, 100% of its full capacity is an example of a multi-state system that has ordered multiple states [7]. A nuclear reactor system [8] or a pumping system [9] which performs differently according to the many different combinations of states of its subsystems are examples of multi-state systems with unordered multiple states. A light-emission diode which emits red, green, and yellow lights under different inputs is an example of a multi-state system in which each of its states has binary values. Furthermore, a state in a system may take a continuous range of quantitative measurement instead of discrete levels, for example, a braking system might produce an output braking force ranging from 250 to 300 kilograms.

Some extension of importance measures from binary systems to multi-state systems has been done. Reference [10] extends the [11] concept of reliability importance of system components to multi-state monotone systems, while [12]–[17] mainly focus on how to extend Birnbaum's importance in binary systems to multi-state systems.

Very few publications discuss how the particular states of a component contribute to a multi-state system, and how the presence of a component and a particular state of a component affect the contributions of other components in the system. Such an investigation has theoretical importance as well as practical value, because the knowledge gained enables efficient design of the system, as well as formulation of optimal strategies for maintaining the system. In a binary system, reliability optimization mainly deals with maximizing the system reliability under constraints such as cost, weight, and/or size, or on minimizing the cost under reliability constraints. This optimization task is by no means trivial, unless the system is very simple. In multi-state systems where components have more than 2 states and the performance utility of the system is to be maximized, the optimization task is obviously more difficult.

Section II introduces a new utility-importance for measuring the contribution of various component-states to the system-performance, and compares this utility-importance to Griffith's importance. Section III studies the contribution of multi-state components to the system; examples are given to show that, for a particular system, even if a multi-state component produces better system-performance than another component, the reverse might happen in another system. Section IV discusses optimization of the system-performance utility by selection and permutation of components, and an application of heuristic methods. Section V gives an example to illustrate the use of heuristic algorithms in optimization of a multi-state system.

II. State Performance Utility

Assumptions:

1) A multi-state system has a zero state and K nonzero states.
2) Each component has a zero state and M nonzero states.

Most importance-concepts have been built on how change-in-state of one component affects the system [12]–[17]. Reference [17, p. 741] proposes the Griffith *importance vector*, $\mathbf{I}_i^G$, to study component #i ($i = 1, \ldots, n$) in a multi-state system:

$$\begin{aligned}\mathbf{I}^G(i) &= \left(I_1^G(i), \ldots, I_M^G(i)\right),\\ I_m^G(i) &= \sum_{j=1}^{K}(a_j - a_{j-1})\\ &\quad\cdot[\Pr\{\phi(m_i, \mathbf{X}) \geq j\} - \Pr\{\phi((m-1)_i, \mathbf{X}) \geq j\}],\\ &\quad m = 1, \ldots, M. \end{aligned} \tag{1}$$

The $I_m^G(i)$ in Griffith's importance vector can be interpreted as the hange of the system performance when component i deteriorates from state m to state $m-1$; it can be regarded as the importance of state m of system-component #i.

A drawback of $\mathbf{I}^G(i)$ is that it measures only how the change of a particular component affects the system performance, but does not measure which component affects it the most, or which state of a certain component contributes the most. However, the extent to which a component and its states affect the system is a major concern to the system designer and the system controller. To overcome this deficiency of $\mathbf{I}^G(i)$, a new performance utility importance function $I_m^{\mathrm{UI}}(i)$ is introduced in *definition* 2.1.

Definition 2.1:

$$\begin{aligned}I_m^{\mathrm{UI}}(i) &\equiv \sum_{j=0}^{K} a_j \cdot \Pr\{\phi(\mathbf{X}) = j,\, \mathbf{X}_i = m\}\\ &= \sum_{j=0}^{K} a_j \cdot \Pr\{\phi(\mathbf{X}) = j | \mathbf{X}_i = m\} \cdot \Pr\{\mathbf{X}_i = m\}\\ &= p_{i,m} \cdot \sum_{j=0}^{K} a_j \cdot \Pr\{\phi(m_i, \mathbf{X}) = j\}. \end{aligned} \tag{2}$$

$I_m^{\mathrm{UI}}(i)$ can be interpreted as the contribution of state m of component i to the system. Thus the utility importance of component i can be defined as the vector:

$$\mathbf{I}^{\mathrm{UI}}(i) = \left(I_0^{\mathrm{UI}}(i), I_1^{\mathrm{UI}}(i), \ldots, I_M^{\mathrm{UI}}(i)\right). \tag{3}$$

Straight-forward calculation gives the following relationship between $I_m^{\mathrm{UI}}(i)$ and coordinate $I_m^G(i)$:

$$\frac{\partial I_m^{\mathrm{UI}}(i)}{\partial p_{i,m}} - \frac{\partial I_{m-1}^{\mathrm{UI}}(i)}{\partial p_{i,m-1}} = I_m^G(i), \qquad m = 1, \ldots, M.$$

A significance of $I_m^{\mathrm{UI}}(i)$ is that for each $i = 1, \ldots, n$, the U of a system (which is a measure of the overall effect of all components in the system) can be expressed in terms of $I_m^{\mathrm{UI}}(i)$:

$$\begin{aligned}U &= \sum_{j=0}^{K} a_j \cdot \Pr\{\phi(\mathbf{X}) = j\}\\ &= \sum_{j=0}^{K} a_j \cdot \left[\sum_{m=0}^{M} \Pr\{\phi(\mathbf{X}) = j | \mathbf{X}_i = m\} \cdot \Pr\{\mathbf{X}_i = m\}\right]\\ &= \sum_{m=0}^{M}\sum_{j=0}^{K} a_j \cdot \Pr\{\phi(m_i, \mathbf{X}) = j\} \cdot p_{i,m},\\ &= \sum_{m=0}^{M} I_m^{\mathrm{UI}}(i). \end{aligned} \tag{4}$$

Equation (4) shows that a state m of component i with larger $I_m^{\mathrm{UI}}(i)$ contributes appreciably more to the system performance utility. To illustrate how a system is described differently using $I_m^{\mathrm{UI}}(i)$ and $I_m^G(i)$, consider a 1-component system, in which $\mathbf{X} = (\mathbf{X}_1)$, with 4 states $j = 0, 1, 2, 3$ and structure function $\phi(\mathbf{X}_1 = j) = \phi(j) = j$, $j = 0, 1, 2, 3$. Let the system component have 4 states $m = 0, 1, 2, 3$ with probability distribution $(p_{1,0}, p_{1,1}, p_{1,2}, p_{1,3}) = (0.1, 0.1, 0.7, 0.1)$. Table I shows $I_m^{\mathrm{UI}}(1)$ ($m = 0, 1, 2, 3$) and $I_m^G(1)$ ($m = 1, 2, 3$) calculated for 2 sets of values of a_0, a_1, a_2, a_3. The $I_m^G(1)$ is not

TABLE I
VALUES OF $I_1^{\mathrm{UI}}(m)$ AND $I_1^G(m)$ FOR 2 SETS OF (a_0, a_1, a_2, a_3)

	$m=0$	$m=1$		$m=2$		$m=3$	
Performance utility level	$I_1^{\mathrm{UI}}(0)$	$I_1^{\mathrm{UI}}(1)$	$I_1^G(1)$	$I_2^{\mathrm{UI}}(1)$	$I_2^G(1)$	$I_3^{\mathrm{UI}}(1)$	$I_3^G(1)$
$(a_0,a_1,a_2,a_3)=(0,100,1000,8000)$	0	10	100	700	900	800	7000
$(a_0,a_1,a_2,a_3)=(0,100,3000,8000)$	0	10	100	2100	2900	800	5000

defined for $m = 0$. For example, when $(a_0, a_1, a_2, a_3) = (0, 100, 1000, 8000)$, then

$$\begin{aligned} I_2^{\mathrm{UI}}(1) &= p_{1,2} \cdot \sum_{j=1}^{3} a_j \cdot \Pr\{\phi(2) = j\} \\ &= 0.7 \cdot (100 \cdot 0 + 1000 \cdot 1 + 8000 \cdot 0) = 700, \\ I_3^G(1) &= (a_1 - a_0) \cdot (\Pr\{\phi(3) \geq 1\} - \Pr\{\phi(2) \geq 1) \\ &\quad + (a_2 - a_1) \cdot (\Pr\{\phi(3) \geq 2\} - \Pr[\phi(2) \geq 2\}) \\ &\quad + (a_3 - a_2) \cdot (\Pr\{\phi(3) \geq 3\} - \Pr\{\phi(2) \geq 3]) \\ &= (100 - 0) \cdot (1 - 1) + (1000 - 100) \cdot (1 - 1) \\ &\quad + (8000 - 1000) \cdot (1 - 0) = 7000, \ldots. \end{aligned}$$

Even though state 2 of the component has the highest probability of occurrence ($p_{1,2} = 0.7$), Table I shows that when $a_2 = 1000$, then state 2 does not always have the highest contribution to the system, because $I_2^{\mathrm{UI}}(1) = 700 < I_3^{\mathrm{UI}}(1) = 800$. When $a_2 = 1000$, from the user's view-point, effort should be made to keep the component at state 3, because

$$I_3^{\mathrm{UI}}(1) = 800 > I_2^{\mathrm{UI}}(1) = 700 > I_1^{\mathrm{UI}}(1) = 10.$$

On the other hand, when $a_2 = 3000$, effort should be made to keep the component at state 2, because for this case,

$$I_2^{\mathrm{UI}}(1) = 2100 > I_3^{\mathrm{UI}}(1) = 800 > I_1^{\mathrm{UI}}(1) = 10.$$

Even though state 3 has a $p_{1,3}$ (=0.1) smaller than $p_{1,2}$ and equal to $p_{1,0}$ and $p_{1,1}$, Table I shows that changing the component from state 3 to state 2 (rather than changing it from state 2 to state 1, or from state 1 to state 0) has the largest effect on the system as measured by Griffith's importance, because $I_3^G(1) > I_2^G(1) > I_1^G(1)$ for both $a_2 = 1000$ and $a_2 = 3000$.

Definition 2.2 [18]: A system is a

a) multi-state *series system* if and only if its structure function is $\phi(X) = \min[x_1, x_2, \ldots, x_n]$,
b) multi-state *parallel system* if and only if its structure function is $\phi(X) = \max[x_1, x_2, \ldots, x_n]$.

Proposition 2.1: Notation

$$\Psi_{n,j}(i) \equiv \prod_{\substack{k=1 \\ k \neq i}}^{n} R_{k,j}.$$

The utility importance of state m $(0 \leq m \leq M)$ of component i in a series system is

$$I_m^{\mathrm{UI}}(i) = p_{i,m} \cdot \left[\sum_{j=0}^{m-1} a_j \cdot (\Psi_{n,j}(i) - \Psi_{n,j+1}(i)) + a_m \cdot \Psi_{n,m}(i) \right] \quad (5)$$

for $m > 0$, and

$$I_m^{\mathrm{UI}}(i) = p_{i,m} \cdot [a_m \cdot \Psi_{k,n}] = a_0 \cdot p_{i,0}, \quad (6)$$

for $m = 0$.

Proof: For this series system,

$$\begin{aligned} \Pr\{\phi(\mathbf{X}) \geq j\} &= \Pr\{\mathbf{X}_1 \geq j, \mathbf{X}_2 \geq j, \ldots, \mathbf{X}_n \geq j\} \\ &= \prod_{k=1}^{n} R_{j,k}. \end{aligned} \quad (7)$$

By definition,

$$I_m^{\mathrm{UI}}(i) = p_{i,m} \cdot \sum_{j=0}^{K} a_j \cdot \Pr\{\phi(m_i, \mathbf{X}) = j\}.$$

When $0 < m < M$, the sum in $I_m^{\mathrm{UI}}(i)$ can be written as

$$\sum_{j=0}^{M} = \sum_{j=0}^{m-1} + \sum_{j=m} + \sum_{j=m+1}^{M}.$$

The $\sum_{j=m}$ consists of a single term, while for a series system each term in $\sum_{j=m+1}^{M}$ is 0 because $\Pr\{\phi(m_i, \mathbf{X}) = j\} = 0$ for all $j > m$. Hence

$$\begin{aligned} &I_m^{\mathrm{UI}}(i) \\ &= p_{i,m} \cdot \left[\sum_{j=0}^{m-1} a_j \cdot \Pr\{\min[\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_n] = j\} \right. \\ &\quad \left. + a_m \cdot \Pr\{\min[\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_n] \geq m\} \right] \\ &= p_{i,m} \cdot \left[\sum_{j=0}^{m-1} a_j \cdot (\Psi_{n,j}(i) - \Psi_{n,j+1}(i)) + a_m \cdot \Psi_{n,m}(i) \right]. \end{aligned} \quad (8)$$

This proves proposition 2.1 for $0 < m < M$. The proofs for the other cases are similar, except that the sum $\sum_{j=0}^{M}$ is broken in slightly different ways:

$$\text{For } 0 < m = M, \text{ use } \sum_{j=0}^{M} = \sum_{j=0}^{m-1} + \sum_{j=m}.$$

$$\text{For } 0 = m < M, \text{ use } \sum_{j=0}^{M} = \sum_{j=m} + \sum_{j=m+1}^{M}.$$

End of Proof for Proposition 2.1.

Proposition 2.2: Notation

$$\Psi^*_{n,\,j}(i) \equiv \prod_{\substack{k=1\\k\neq i}}^{n} (1 - R_{k,\,j}).$$

The utility importance of state m, $0 \le m \le M$ of component i in a parallel system is

$$I^{\mathrm{UI}}_m(i) = p_{i,\,m} \cdot \left[a_m \cdot \Psi^*_{n,\,m+1}(i) + \sum_{j=m+1}^{M} a_j \cdot \left(\Psi^*_{n,\,j+1}(i) - \Psi^*_{n,\,j}(i)\right) \right] \quad \text{for } m < M, \tag{9}$$

$$I^{\mathrm{UI}}_m(i) = p_{i,\,m} \cdot \left(a_m \cdot \Psi^*_{n,\,m+1}(i)\right) = p_{i,\,M} \cdot a_M, \quad \text{for } m = M. \tag{10}$$

Remark: In $\Psi^*_{n,\,m+1}(i)$ in (10), $R_{k,\,m+1} = 0$ when $m = M$ because the highest state of all components is M.

Proof: For a parallel system,

$$\Pr\{\phi(\mathbf{X}) \ge j\} = \Pr\{\max[x_1, x_2, \ldots, x_n] \ge j\} = 1 - \prod_{k=1}^{n} (1 - R_{k,\,j}). \tag{11}$$

When

$$0 < m < M, \qquad \sum_{j=0}^{M} = \sum_{j=0}^{m-1} + \sum_{j=m} + \sum_{j=m+1}^{M};$$

for a parallel system, each term in the sum $\sum_{j=0}^{m-1}$ is 0 because $\Pr\{\phi(m_i, \mathbf{X}) = j\} = 0$ for all $j < m$; thus

$$\begin{aligned} I^{\mathrm{UI}}_m(i) &= p_{i,\,m} \cdot \sum_{j=0}^{M} a_j \cdot \Pr\{\phi(m_i, \mathbf{X}) = j\} \\ &= p_{i,\,m} \cdot a_m \cdot \Pr\{\max[\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_n] \le m\} \\ &\quad + p_{i,\,m} \cdot \sum_{j=m+1}^{M} a_j \cdot \Pr\{\max[\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_n] = j\} \\ &= p_{i,\,m} \cdot \left[a_m \cdot \Psi^*_{n,\,m+1}(i) + \sum_{j=m+1}^{M} a_j \cdot \left(\Psi^*_{n,\,j+1}(i) - \Psi^*_{n,\,j}(i)\right) \right]. \end{aligned} \tag{12}$$

This proves proposition 2.2 for $0 < m < M$. The proofs for other cases are similar, except that $\sum_{j=0}^{M}$ is broken up in slightly different ways:

$$\text{For } 0 = m < M\text{: } \sum_{j=0}^{M} = \sum_{j=m} + \sum_{j=m+1}^{M}.$$

$$\text{For } 0 < m = M\text{: } \sum_{j=0}^{M} = \sum_{j=0}^{m-1} + \sum_{j=m}.$$

III. Component Performance Utility in Multi-State Systems

In general, there are 2 ways to improve the reliability of a binary system: 1) increase the reliability of individual components, and/or 2) add redundant components to the system. For a multi-state system, however, the situation is not so simple, and few publications discuss how the performance utility of a multi-state system can be improved. It is not even straightforward to define how a multi-state component can be regarded as better than another one. If a position in a multi-state system, e.g., k, can be occupied by either component i_1 or component i_2, then the index $I(k, i_1, i_2)$ can be used to compare the contributions of components i_1 and i_2 to the performance utility of this system:

$$\begin{aligned} &I(k, i_1, i_2) \\ &= \sum_{j=1}^{M} a_j \cdot [\Pr\{\phi(\mathbf{X}) = j | \psi_k(i_1)\} - \Pr\{\phi(\mathbf{X}) = j | \psi_k(i_2)\}] \\ &\qquad (\psi_k(i) \equiv \text{position } k \text{ is occupied by component } i). \end{aligned} \tag{13}$$

Component i_1 can be regarded as better than component i_2 for position k of a particular system if $I(k, i_1, i_2) > 0$, *vice versa* if $I(k, i_1, i_2) < 0$, and both components can be regarded as equally good if $I(k, i_1, i_2) = 0$. However, without regard to a particular system and a particular position in the system, it is hard to define whether one component is better than another. This is illustrated by the example:

Four components (1, 2, 3, 4), each have 3 states; let

$$\begin{aligned} (p_{1,\,0}, p_{1,\,1}, p_{1,\,2}) &= (0.2, 0.5, 0.3), \\ (p_{2,\,0}, p_{2,\,1}, p_{2,\,2}) &= (0.3, 0.1, 0.6), \\ (p_{3,\,0}, p_{3,\,1}, p_{3,\,2}) &= (0.2, 0.4, 0.4), \\ (p_{4,\,0}, p_{4,\,1}, p_{4,\,2}) &= (0.4, 0.5, 0.1). \end{aligned}$$

Let a series system consist of 2 components; thus $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$, and $\phi(X) = \min[x_1, x_2]$, where x_1, x_2 can take the values 0, 1, 2, and position 1 in the system can be occupied by either component 1 or component 2, and position 2 can be occupied by either component 3 or component 4.

Let $U(i_1, i_2) \equiv$ performance utility of this system when position 1 is occupied by component i_1, $i_1 = 1, 2$ and position 2 is occupied by component i_2, $i_2 = 3, 4$.

Values of $U(i_1, i_2)$ for 2 sets of values of performance utility levels are calculated and shown in Table II. For example, when $(a_0, a_1, a_2) = (0, 1000, 2000)$, then

$$\begin{aligned} U(1, 3) &= a_0 \cdot \Pr\{\phi(\mathbf{X}) = 0\} + a_1 \cdot \Pr\{\phi(\mathbf{X}) = 1\} + a_2 \cdot \Pr\{\phi(\mathbf{X}) = 2\} \\ &= 0 \cdot \Pr\{\min[x_1, x_2] = 0\} + 1000 \cdot \Pr\{\min[x_1, x_2] = 1\} + 2000 \cdot \Pr\{\min[x_1, x_2] = 2\} \\ &= 1000 \cdot (p_{1,\,1} \cdot p_{3,\,1} + p_{1,\,1} \cdot p_{3,\,2} + p_{1,\,2} \cdot p_{3,\,1}) + 2000 \cdot (p_{1,\,2} \cdot p_{3,\,2}) \\ &= 520 + 240 = 760, \end{aligned}$$

and so on.

TABLE II
VALUES OF $U(i_1, i_2)$ FOR 2 SETS OF (a_0, a_1, a_2, a_3)

Performance utility level	$U(1,3)$	$U(2,3)$	$U(1,4)$	$U(2,4)$
$(a_0, a_1, a_2) = (0, 1000, 2000)$	760	800	510	480
$(a_0, a_1, a_2) = (0, 1400, 2000)$	968	928	690	624

Table II shows that

$$U(1,3) - U(2,3) < 0 \quad \text{when } (a_1, a_2) = (1000, 2000),$$
$$U(1,3) - U(2,3) > 0 \quad \text{when } (a_1, a_2) = (1400, 2000).$$

Thus the order between $U(1,3)$ and $U(2,3)$ can change if performance utility levels change. Therefore, in general it is not possible to tell whether component 1 is better than component 2 or *vice versa,* if the performance utility levels are not known.

Table II also shows that

$$U(1,3) - U(2,3) > 0 \quad \text{and} \quad U(1,4) - U(2,4) > 0$$

when $(a_1, a_2) = (1400, 2000)$;

$$U(1,3) - U(2,3) < 0 \quad \text{and} \quad U(1,4) - U(2,4) > 0$$

when $(a_1, a_2) = (1000, 2000)$.

This means that for the same system, for certain performance utility levels, replacing a component with another component can increase the system performance utility, while for other performance utility levels, this could decrease the system performance level. This means that it is not possible to know whether one component is better another one, if the utility levels of the system are not known. Thus, this example shows that if one component in a system is replaced with a spare component, it is not possible to tell in general whether this replacement will increase or decrease the performance utility of the system. In other words, by ignoring the performance utility levels and the probability distribution of the components, it is impossible to define a meaningful index to measure the performance utility of an individual component.

IV. MAXIMIZATION OF PERFORMANCE UTILITY

Sections II and III investigate only the effect of individual components on the system; this section considers the problem of placement of components to optimize the system.

Assumptions:

1) N multi-state components are available.
2) Each of these components can perform the same function.
3) These components have different probability distributions of states.
4) A system requires $n(<N)$ components.
5) The optimization problem for the system is to select n of the N components, and define the position order of the selected components to maximize the performance utility of the system.

For a binary system, this process has 2 steps:

1) Select n of the N components with the highest reliability.
2) Define their position order in the system so that the performance utility of the system is maximized.

See [19]–[21], for example, on optimization of particular types of systems. For a multi-state system, the optimization process cannot be performed in these 2 discrete steps, because, as shown in Section III, comparison of the performance utility of 2 multi-state components with known probability distributions of states is not trivial, and it is not straightforward to define how one component can be regarded as having higher reliability than another.

In an n-component system, the total number of ways of selecting and arranging n of the N components is $N \cdot (N-1) \cdots (N-n+1)$, which can be as large as 3 991 680, when $N = 12$, and $n = 7$. Hence, even when n and N are of moderate sizes, calculation of the performance utility of the system for all possible permutations is time-consuming and impractical.

Similar to the binary case, special algorithms can be established for optimizing multi-state systems. Construction of such algorithms is a separate topic of research and is not treated in this paper. These days, because high-speed computing machines and efficient heuristic algorithms are readily available, using these heuristic methods for optimization of the performance utility U is practical and efficient. Commonly used heuristic methods such as GA, SA, and TS can be used for optimizing multi-state systems. See [22]–[27] for studies of these search algorithms and [28], [29]. References [30]–[37] for their applications in reliability analysis and other areas. Many other heuristic methods, such as Artificial Neural Networks [38] and Threshold Accepting [39] can also be applied, but they are not discussed here.

To search for a global maximum of an objective function $f(x)$ over a search space using GA, each element x in the search space is represented as a string of symbols called: chromosome. Each chromosome in the search space corresponds to a unique x and has a fitness-value, $f(x)$. The search process begins with a population which is a subset of the search space. A mating-pool is formed by selecting chromosomes with replacement from the population in such a way that a chromosome x is selected with probability $f(x)/\sum_y f(y)$, where $\sum_y$ runs over the entire population. More precisely, a random number $p \in (0, 1)$ is generated, and x is accepted or rejected according to whether $p \le f(x)/\sum_y f(y)$ or $p > f(x)/\sum_y f(y)$.

Next, pairs of chromosomes are selected from the mating pool according to a pre-determined probability. In each selected pair, the 2 chromosomes are then divided randomly into sub-strings in the same way, and sub-strings from the 2 chromosomes are swapped to form 2 new chromosomes called "offspring." This process is called "crossover." The fitness values of the offspring are calculated, and a new population is formed by replacing parent chromosomes of low fitness values in the mating pool with offspring of high fitness values. The content of sub-strings of chromosomes in the population are then selected with a small probability and altered (mutated), to avoid $f(x)$ converging to a local maximum. These steps are repeated until a certain stopping rule is satisfied.

To search for a global minimum of a function $\Phi(x)$ using SA, an initial point, x_0, is selected from the search space. Then another point, x_1, within a certain small distance from x_0 is selected randomly. The point x_1 is accepted with probability

TABLE III
PROBABILITY DISTRIBUTION OF THE STATES OF 60 COMPONENTS

#	State 0	State 1	State 2	#	State 0	State 1	State 2	#	State 0	State 1	State 2
1	0.12	0.10	0.78	21	0.16	0.07	0.77	41	0.12	0.12	0.76
2	0.24	0.13	0.63	22	0.20	0.14	0.66	42	0.16	0.24	0.60
3	0.40	0.37	0.23	23	0.30	0.40	0.30	43	0.22	0.40	0.38
4	0.21	0.24	0.55	24	0.20	0.21	0.59	44	0.38	0.06	0.56
5	0.26	0.01	0.73	25	0.09	0.20	0.71	45	0.06	0.20	0.74
6	0.15	0.52	0.33	26	0.38	0.31	0.31	46	0.06	0.60	0.34
7	0.04	0.66	0.30	27	0.23	0.40	0.37	47	0.11	0.50	0.39
8	0.05	0.26	0.69	28	0.15	0.15	0.70	48	0.21	0.11	0.68
9	0.07	0.73	0.20	29	0.32	0.50	0.18	49	0.56	0.22	0.22
10	0.13	0.21	0.66	30	0.29	0.08	0.63	50	0.09	0.30	0.61
11	0.12	0.23	0.65	31	0.28	0.10	0.62	51	0.21	0.15	0.64
12	0.14	0.02	0.84	32	0.10	0.05	0.85	52	0.08	0.10	0.82
13	0.05	0.85	0.10	33	0.10	0.75	0.15	53	0.12	0.70	0.18
14	0.11	0.01	0.88	34	0.13	0.01	0.86	54	0.10	0.03	0.87
15	0.07	0.30	0.63	35	0.18	0.20	0.62	55	0.15	0.20	0.65
16	0.08	0.43	0.49	36	0.30	0.25	0.45	56	0.27	0.27	0.46
17	0.25	0.23	0.52	37	0.10	0.35	0.55	57	0.40	0.02	0.58
18	0.10	0.31	0.59	38	0.26	0.17	0.57	58	0.30	0.16	0.54
19	0.35	0.14	0.51	39	0.09	0.01	0.90	59	0.30	0.18	0.52
20	0.19	0.10	0.71	40	0.10	0.20	0.70	60	0.20	0.07	0.73

1 if $\Delta\Phi = \Phi(x_1) - \Phi(x_0) \leq 0$, and accepted with probability $\exp[-\alpha \cdot \Delta\Phi]$ if $\Delta\Phi > 0$, where α is a positive constant. These steps are repeated until a certain stopping rule is satisfied. In maximizing the U of a system with SA, on can set $\Phi = -U$.

The search for a global maximum using TS begins with an initial solution. At each step, a move is performed between components in the solution of the previous step and a component in the "spare component store" which is a subset $S \backslash T$ of the search space S. Here T is a TS containing some previous moves. The purpose of excluding moves in the TS is to prevent the algorithm from converging to a local maximum. However, an "improved best aspiration criterion" is adopted, so that if a certain move contained in the TS improves the objective function, then tabu classification of this move can be "overridden"; and this "aspiration-level move" can be accepted. At each step, the move is chosen so that the objective function gains the largest increase (which at times could be negative). The process terminates when a certain stopping rule is satisfied.

These 3 algorithms have various modified versions, e.g., in [24], [25]. Section V gives a simple example illustrate how a multi-state system can be optimized using these algorithms.

V. AN EXAMPLE

Consider a 3-state system consisting of $n = 4$ 3-state components, such that the system is at state 0 when at least 2 of its components are at state 0, state 2 when at least 3 consecutive components are at state 2, state 1 otherwise.

The search space consists of $N = 60$ components; Table III gives their state probability distributions.

Let $(a_0, a_1, a_2) = (0, 2500, 3000)$. Then

$$U = 0 \cdot \Pr\{\Phi(\mathbf{X}) = 0\} + 2500 \cdot \Pr\{\Phi(\mathbf{X}) = 1\} + 3000 \cdot \Pr\{\Phi(\mathbf{X}) = 2\}.$$

The arrangement in which x_i is occupied by component s_i is represented by $s_1|s_2|s_3|s_4$; e.g., $13|7|54|29$ represents:

position 1 is occupied by component 13,
position 2 is occupied by component 7,
position 3 is occupied by component 54,
position 4 is occupied by component 29.

In this example, to optimize the performance utility of the system using GA, 20 components are selected from a total of 60, to form an initial population. A chromosome is represented by $s_1|s_2|s_3|s_4$. Pairs in the mating pool are selected with probability 0.6 for crossover. A chromosome in the population is selected with probability 0.1 for mutation, in which 1 of the 4 components in the chromosome is selected randomly and replaced by another component chosen randomly from the population. Fig. 1 shows that U becomes stable after generation #120 or so. A maximum value $U = 2728.7$ is achieved at the arrangement $7|33|13|9$.

When SA is used in this example, a randomly selected 4-component set is used as the initial solution. The distance between 2 components with probability distributions $(p_{1,0}, p_{1,1}, p_{1,2})$ and $(p_{2,0}, p_{2,1}, p_{2,2})$ for states 0, 1, 2 is defined as

$$\frac{p_{1,0} \cdot p_{2,0} + p_{1,1} \cdot p_{2,1} + p_{1,2} \cdot p_{2,2}}{[(p_{1,0}^2 + p_{1,1}^2 + p_{1,2}^2) \cdot (p_{2,0}^2 + p_{2,1}^2 + p_{2,2}^2]^{1/2}}.$$

A neighborhood-set of a component is the collection of the 5 components that are closest in distance to it. In each step, each component in the system is replaced with probability 1 with a component selected randomly from its neighborhood if this replacement gives a lower value $\Phi_1 = -U_1$ of the negative of system performance utility than the original value $\Phi_0 = -U_0$; otherwise, the component is replaced with probability $\exp[\alpha(\Phi_1 - \Phi_0)]$, where the initial value of α is chosen as 1/12, and the value of α in each step is the previous value

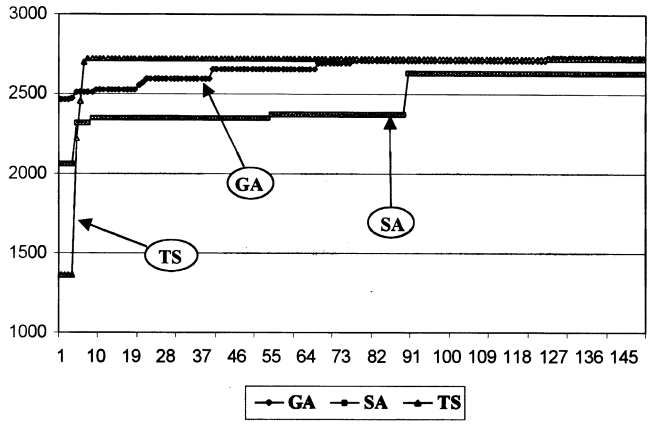

Fig. 1. The results of optimizing a system using GA, SA, TS. The horizontal axis is the number of iterations; the vertical axis is the system performance utility.

multiplied by 0.9. Here the values 1/12 and 0.9 are chosen by experience in an *ad hoc* manner; the algorithm converges reasonably quickly under this condition. Fig. 1 shows that U stabilizes after the 90th iteration or so, and a maximum value of 2626.32 is achieved at the arrangement $47|46|9|13$.

When TS is used, a move is defined as replacement of a component in the system in the previous step of iteration by a component in the spare component store. In each step, the tabu set T consists of the moves performed within the previous 3 steps. The tabu status is overridden for a move if the objective function after this move is larger than all previous values. The initial solution used in this example is randomly selected. Fig. 1 shows that the algorithm stabilizes at iteration #9. A maximum value, 2723.31, of U is reached at the arrangement $46|13|9|7$.

In this example, GA converges more steadily than SA, and TS converges appreciably faster than GA and SA. For the values of the maximum performance utility obtained, GA is slightly better than TS, and TS is better than SA. This example shows that various heuristic methods can perform differently when they are applied to the same multi-state system. Comparative study of the performance of GA, SA, TS, and other heuristic methods in optimization of multi-state systems is a separate topic of investigation, and is not handled in this paper.

ACKNOWLEDGMENT

The authors would like to thank the referees for their valuable comments and suggestions that appreciably improved the quality of this paper.

REFERENCES

[1] E. A. Elsayed, *Reliability Engineering*: Addison Wesley Longman, 1996.
[2] A. Høyland and M. Rausand, *System Reliability Theory: Models and Statistical Methods*: John Wiley & Sons, 1994.
[3] F. C. Meng, "Comparing the importance of system components by some structural characteristics," *IEEE Trans. Rel.*, vol. 45, no. 1, pp. 59–65, 1996.
[4] ——, "Some further results on ranking the importance of system components," *Reliab. Eng. Syst. Saf.*, vol. 47, pp. 97–101, 1995.
[5] J. S. Hong and C. H. Lie, "Joint reliability-importance of two edges in an undirected network," *IEEE Trans. Rel.*, vol. 42, no. 1, pp. 17–23, 1993.
[6] M. J. Armstrong, "Joint reliability-importance of components," *IEEE Trans. Rel.*, vol. 44, no. 3, pp. 408–412, 1995.
[7] A. P. Wood, "Multistate block diagrams and fault trees," *IEEE Trans. Rel.*, vol. R-34, no. 3, pp. 236–240, 1985.
[8] S. Garribba, E. Guagnini, and P. Mussio, "Multistate block diagrams and fault trees," *IEEE Trans. Rel.*, vol. R-34, no. 5, pp. 463–472, 1985.
[9] A. Gandini, "Importance & sensitivity analysis in assessing system reliability," *IEEE Trans. Rel.*, vol. 39, no. 1, pp. 61–69, 1990.
[10] V. C. Bueno, "On the importance of components for multi-state monotone systems," *Statist. Prob. Lett.*, vol. 7, pp. 51–59, 1989.
[11] R. E. Barlow and F. Proschan, "Importance of system components and fault tree analysis," Operation Research Center, University of California, Berkeley, ORC-74-3, 1974.
[12] R. E. Barlow and A. S. Wu, "Coherent systems with multi-state components," *Math. Oper. Res.*, vol. 3, pp. 275–281, 1978.
[13] T. Aven, "On performance measures for multi-state monotone systems," *Reliab. Eng. Syst. Saf.*, vol. 41, pp. 259–266, 1993.
[14] B. Natvig, "Two suggestions of how to define a multi-state coherent system," *Adv. Appl. Prob.*, vol. 14, pp. 434–457, 1982.
[15] H. W. Block, "A decomposition for multi-state monotone system," *J. Appl. Prob.*, vol. 19, pp. 391–402, 1982.
[16] F. C. Meng, "Component-relevancy and characterization results in multi-state systems," *IEEE Trans. Rel.*, vol. 42, no. 3, pp. 478–483, 1993.
[17] W. S. Griffith, "Multi-state reliability models," *J. Appl. Prob.*, vol. 17, pp. 735–744, 1980.
[18] E. El-Neweihi, F. Proschan, and J. Sethuraman, "Multi-state coherent systems," *J. Appl. Prob.*, vol. 15, pp. 675–688, 1978.
[19] P. J. Boland, F. Proschan, and Y. L. Tong, "Optimal arrangement of components via pairwise rearrangements," *Nav. Res. Logist.*, vol. 36, pp. 807–815, 1989.
[20] J. Xue and K. Yang, "Symmetric relations in multi-state systems," *IEEE Trans. Rel.*, vol. 44, no. 4, pp. 689–693, 1995.
[21] M. Zuo and W. Kuo, "Design and performance analysis of consecutive-k-out-of-n structures," *Nav. Res. Logist.*, vol. 37, pp. 203–230, 1990.
[22] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 15, no. 5, pp. 549–533, 1986.
[23] P. L. Hammer, Ed., "Linkages with artificial intelligence," in *Ann. Oper. Res.*, 1989, vol. 21, complete volume.
[24] C. R. Reeves, *Modern Heuristic Techniques for Combinational Problems*: John Wiley, 1993.
[25] H. Youssef, S. M. Sait, and H. Adiche, "Evolutionary algorithms, simulated annealing and tabu search: A comparative study," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 2, pp. 167–181, 2001.
[26] W. Banzhaf *et al.*, Ed., *Proc. Genetic and Evolutionary Computation Conf. (GECCO-1999)*: Morgan Kaufmann Publishers, 1999, vol. 1 and 2.
[27] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, "Generalized simulated annealing for function optimization," *Technometrics*, vol. 28, no. 3, pp. 209–217, 1986.
[28] N. Metropolis *et al.*, "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, 1953.
[29] R. R. Brooks and S. S. Iyengar, *Multi-Sensor Fusion: Fundamentals and Applications with Software*: Prentice Hall, 1998.
[30] A. Kumar, R. M. Pathak, and Y. P. Gupta, "Genetic-algorithm-based reliability optimization for computer network expansion," *IEEE Trans. Rel.*, vol. 44, no. 1, pp. 63–72, 1995.
[31] L. Painton and J. Campbell, "Genetic algorithm in optimization of system reliability," *IEEE Trans. Rel.*, vol. 44, no. 2, pp. 172–178, 1995.
[32] D. W. Coit and A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Trans. Rel.*, vol. 45, no. 2, pp. 254–260, 1996.
[33] S. R. V. Majety and J. Rajgopal, "Dynamic penalty function for evolutionary algorithms with an application to reliability allocation," in *Industrial Engineering Research—Conf. Proc.*, 1997, pp. 36–41.
[34] V. Ravi, B. S. N. Murty, and P. J. Reddy, "Nonequilibrium simulated annealing-algorithm applied to reliability optimization of complex systems," *IEEE Trans. Rel.*, vol. 46, no. 2, pp. 233–239, 1997.
[35] J. E. Angus and K. Ames, "Simulated annealing algorithm for system cost minimization subject to reliability constraints," *Communications in Statistics, Part B: Simulation and Computation*, vol. 26, no. 2, pp. 783–790, 1997.
[36] F. Glover and M. Laguna, *Tabu Search*: Kluwer Academic Publications, 1997.
[37] S. Pierre and A. Elgibaoui, "Tabu-search approach for designing computer-network topologies with unreliable components," *IEEE Trans. Rel.*, vol. 46, no. 3, pp. 350–359, 1997.
[38] R. Rojas, *Neural Networks: A Systematic Introduction*: Springer, 1996.

[39] P. Winker and K. T. Fang, "Application of threshold accepting to the evaluation of the discrepancy of a set of points," *SIAM J. Numer. Anal.*, vol. 34, pp. 2028–2042, 1997.

Shaomin Wu received the M.Sc. in 1992 and the Ph.D. in 1995 both in Applied Mathematics, from Southeast University in P.R. China. From 1995 to March 2001, he was working at Automation Research Institute of Baosteel Technology Center in Shanghai, P.R. China. Since then he has been working as a Research Fellow in the Department of Computer Science in the University of Bristol. During 1998 and 1999, he visited The University of Hong Kong and City University of Hong Kong, and collaborated with scholars there on several research projects. His research interests include time-series analysis, statistical process control, system reliability, and data mining.

Ling-Yau Chan has a B.Sc. and an M.Phil. in Mathematics, and a Ph.D. in Statistics. He is an Associate Professor in the Department of Industrial and Manufacturing Systems Engineering at The University of Hong Kong. His research interests include reliability analysis, statistical quality control, experiments with mixtures, design of experiments, and optimal designs. He collaborates with researchers from universities and research organizations worldwide; and has published more than 60 research papers.