Design of Reliable System Based on Dynamic Bayesian Networks and Genetic Algorithm

Dingzhou Cao, Ph.D., ReliaSoft Corporation
Shaobai Kan, Ph.D., CUNY
Yu Sun, Ph.D., Wayne State University

Key Words: Genetic Algorithm, Dynamic Bayesian Networks, Reliability optimization, System reliability modeling

SUMMARY & CONCLUSIONS

Traditional approaches to the design of a reliable system follow system requirement analysis, preliminary design, detail design, and evaluation and redesign phases until a final acceptable design is obtained. However, to achieve a shorter time to market, system reliability concerns should be addressed at the design stage ("design for reliability"). In this paper, we propose a reliability optimization framework based on Dynamic Bayesian Networks (DBN) and Genetic Algorithm (GA) which considers system reliability as a design parameter in design stages and can accelerate the design process of a reliable system. The majority of solution methods for reliability optimization problems are based on simple system structures (series, parallel, or k-out-of-n) without component dependency. In this paper, we extend it to a more complicated system with dynamic behavior. In order to capture the different dynamic behaviors of a system, DBN is used to estimate the system reliability of a potential design. Two basic DBN structures "CHOICE" and "REDUNDANCY" are introduced in this study. GA is developed and integrated into a DBN to find the optimal design. Simulation results show that the integration of GA optimization capabilities with DBN provides a robust, powerful system-design tool. Finally, the proposed method is applied to an example of a cardiac-assist system.

1 INTRODUCTION

Reliability is the ability of a system to operate successfully long enough to complete its assigned mission under stated conditions. Traditional approaches to the design process of a reliable system follow system requirement analysis, preliminary design, detailed design, and evaluation and redesign phases until a final acceptable design is obtained. However, to achieve shorter time to market, system reliability concerns should be addressed at the design stage ("design for reliability"). Thus, the design requirements have to consider reliability, cost, weight, physical size, power consumption, etc. The reliability optimization problem is to select components and redundancy level to optimize some objective function, given system-level constraints on reliability, cost, and/or weight. To deal with this problem, a large number of models and solution methods have been proposed, such as dynamic programming, Lagrangian multiplier, heuristic approaches, and integer programming. Nevertheless, even a simple redundancy allocation problem in a series system with linear constraints is NP-hard; it seems very difficult to solve such a problem efficiently by the traditional methods mentioned above. Recently, Genetic Algorithm (GA) [1], with the advantage of not requiring derivatives, has demonstrated an efficient method to solve this type of problems. GA is powerful for handling many diverse design scenarios, including those with more complex forms of redundancy and mixing of components [2]. For the system involving active and cold-standby redundancy, Coit [3] proposed a genetic algorithm for a redundancy allocation problem for a series-parallel system with active and cold-standby redundancies. However, the existed GA-based methodologies are mainly focus on simple system structure: series, parallel or k-out-of-n with/without simple active or a cold-standby redundancy strategy. There are few studies about the reliability optimization of a complicated system with dynamic behavior. In the paper [4], the author described embedding a GA into an existing fault-tree methodology to determine the heuristic optimal design configuration of a reliable system. She used Dynamic Fault Tree (DFT) to model interactions between events. In addition, [4] used the Markov-based method DIFtree SOLVER to solve the dynamic fault tree. The disadvantage of this method is that, it has a Markov state explosion problem if the system is too complicated. Because of this, in our proposed optimization framework, our model should avoid the problem mentioned above (state explosion); it also must be more practical to model the component dependencies and dynamic failure behaviors between components. Recently, Dynamic Bayesian Networks has shown its powerful ability and flexibility in reliability modeling [5, 6]. Compared with Fault Tree analysis and the Reliability Diagram Blocks method, DBN can handle a multiple state system; compared with Markov Chain-based methods, DBN is more compact and doesn't have a state explosion problem; and most importantly, it can model dynamic system behaviors by using the 2TBN (two time-slice Temporal Bayesian Networks) structure. DBN seems to be the

978-1-4577-1851-9/12/$26.00 ©2012 IEEE

ideal method for system reliability calculations. Furthermore, to date, there has not been any work done on the reliability optimization based on DBN that can handle a complex system with dynamic behaviors. Thus, we will extend the current application of DBN in reliability to a reliability optimization problem by choosing the Dynamic Bayesian Network as the platform and integrating a genetic algorithm into it to build our reliability optimization framework.

The rest of the paper is arranged as follows. Section 2 introduces Dynamic Bayesian Networks and their application in reliability; Section 3 presents the procedure of Genetic Algorithm. Section 4 gives a detailed description of integrating GA and DBN and their application to a numerical example. Finally section 5 concludes the paper.

2 DYNAMIC BAYESIAN NETWORKS (DBN)

2.1 DBN of system reliability modeling

Fault Tree Analysis (FTA) is very powerful when they are solved using binary decision diagrams, but it cannot represent multiple states. The Markov chain can handle multiple states, but the system complexity can induce a state-space explosion problem, which makes the modeling impossible for large system. Because of these difficulties, researchers try to find a more flexible modeling framework for reliability modeling. One such framework that has gained popularity over past decades is the Bayesian Network. The Bayesian Network is a pair $G = ((N, A), D)$, where (N, A) represents the Directed Acyclic Graph (DAG); "N" is a set of nodes; "A" is a set of arcs; and "D" represents the set of probability distributions that are associated with each node. Its main features are a graphical encoding of a set of conditional independence assumptions and a compact way of representing a joint probability distribution between random variables. Dynamic BNs extend the BN formulism by providing an explicit discrete temporal dimension. The standard DBN representation model adopts a discrete time approach, where several time slices are explicated, together with information about transitions from one time slice to the next one. When the Markov assumption holds the future slice at time $t + 1$; it is conditionally independent of the past ones given the present slice at time t. In this case, it is sufficient to represent two consecutive time slices called the anterior and the ulterior layer, and this kind of model is called 2TBN (two time-slice Temporal Bayesian Network) [7]. There is much research on reliability modeling using Dynamic Bayesian Networks (DBN) [5, 6]; however, a detailed discussion of DBN reliability modeling is beyond the scope of this paper.

As for a system design problem respective to reliability optimization, component/subsystem selection and redundancy selection are the two main issues. For simplicity, "CHOICE" and "REDUNDANCY" can be used to denote them. In the following, we will present the DBN structures of "CHOICE" and "REDUNDANCY". Having these, it is easy for us to build a DBN of the whole system by following the method provided by [4].

2.1.1 Component/subsystem selection (CHOICE)

As an example, consider a component/subsystem which has 3 kinds of alternatives for choose. Each potential component alternative has a different failure rate, cost, weight, etc., for example a hard disk (HD) in PC; each potential subsystem has different configuration. To represent this kind of design selection, we introduce the following network structure (Fig. 1) into DBN.

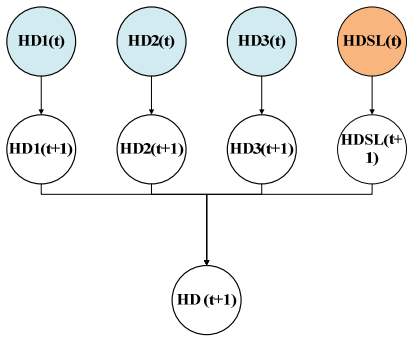

Fig. 1: DBN for CHOICE

The brown node is the selection node, its state=k, $k = [1{:}\,n]$, and n is the number of alternatives. Here $n = 3$. If its state $=k$, it means the kth alternative is selected. For other nodes, let's suppose they have 2 states. 1 means it is working, and 0 means it failed; the failure rate is $\lambda_i^{HD}, i = 1,2,3$. We consider a constant failure rate for options and an exponentially distributed failure time. For node $HDi(t+1)$, it denotes HDi at time slice $t + 1$. Actually, here $t + 1$ node is for capturing the dynamic behavior (PDEP gate, PAND gate, WSP gate, etc.) between components. The probabilities of failure in the network are assigned as (1). (In all the cases not reported, the probability of failure is 0)

$$\begin{cases} P(HD(t+1)=1 \mid HDi(t+1)=1, HDSL(t+1)=i)=1 \\ P(HD(t+1)=0 \mid HDi(t+1)=0, HDSL(t+1)=i)=1 \\ P(HDSL(t+1)=i \mid HDSL(t)=i)=1 \\ P(HDi(t+1)=1 \mid HDi(t)=1)=1-e^{-\lambda_i^{HD}} \\ P(HDi(t+1)=0 \mid HDi(t)=1)=e^{-\lambda_i^{HD}} \end{cases} \tag{1}$$

$$\begin{cases} P(HDi(t+1)=1 \mid HDi(t)=1)=1-e^{-\lambda_i^{HD}} \\ P(HDi(t+1)=0 \mid HDi(t)=1)=e^{-\lambda_i^{HD}} \\ P(HDSL(t+1)=i \mid HDSL(t)=i)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=1, HDSL(t+1)=1)=1 \\ P(HD(t+1)=0 \mid HD1(t+1)=0, HDSL(t+1)=1)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=1, HD2(t+1)=1, HDSL(t+1)=2)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=1, HD2(t+1)=0, HDSL(t+1)=2)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=0, HD2(t+1)=1, HDSL(t+1)=2)=1 \\ P(HD(t+1)=0 \mid HD1(t+1)=0, HD2(t+1)=0, HDSL(t+1)=2)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=1, HD2(t+1)=1, HD3(t+1)=1, HDSL(t+1)=3)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=1, HD2(t+1)=1, HD3(t+1)=0, HDSL(t+1)=3)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=1, HD2(t+1)=0, HD3(t+1)=1, HDSL(t+1)=3)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=0, HD2(t+1)=1, HD3(t+1)=1, HDSL(t+1)=3)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=1, HD2(t+1)=0, HD3(t+1)=0, HDSL(t+1)=3)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=0, HD2(t+1)=1, HD3(t+1)=0, HDSL(t+1)=3)=1 \\ P(HD(t+1)=1 \mid HD1(t+1)=0, HD2(t+1)=0, HD3(t+1)=1, HDSL(t+1)=3)=1 \\ P(HD(t+1)=0 \mid HD1(t+1)=0, HD2(t+1)=0, HD3(t+1)=0, HDSL(t+1)=3)=1 \end{cases} \tag{2}$$

2.1.2 Redundancy selection (REDUNDANCY)

Sometimes, in order to increase system reliability, redundancy is used. Components/subsystems are duplicated in the system design. The network structure to represent this situation is the same as component selection (Fig.1), but the probability assigned to each node is different. For the brown node, its state $=k, k = [1:n]$ where n is the level of redundancy. $k = 1$ means there is only one component, not redundancy; the first component is chosen. $k = 2$ means there are two components. There is one redundancy; the first two components are chosen. $k = n$ means there are $n - 1$ redundancy; the first n components are chosen. The probabilities of failure in the network are assigned as equation (2). (In all cases not reported, the probability of failure is 0).

3 GENETIC ALGORITHM (GA)

A GA is a probabilistic search method for solving an optimization problem. It was first introduced by Holland [1] for effectively handling complex combinatorial problems. A GA maintains a population of different solutions, allowing them to mate, produces offspring, mutate, and fight for survival. The principal of survival of the fittest ensures the population's drive towards optimization. The proposed GA developed for optimizing system design is described in the following subsections.

3.1 GA Operators

GA consists of 3 operators that are used to generate a new population from the previous population:

Reproduction: It is a process to generate a temporary new, auxiliary population; here, population reproduction is performed by implementing the Standard Roulette Selection rule.

Crossover: It involves exchanging the information from two parent individuals and generates two offspring for the next generation; here a one-point crossover is used.

Mutation: It randomly alternates a parameter value on the solution chromosome; it helps maintain sufficient diversity in the population.

3.2 Fitness function

The measure of fitness function is dependent on the problem characteristics. Cost and unreliability are always the two core measures. We will originally use these two in our optimization approach. However, the approach will be flexible to accommodate more criteria, such as power consumption if the system is an electronic product or weight and physical size if the system is a vehicle, and so on. For multiple criteria, we use a weighted fitness function for a trade-off. For these multiple criteria, we have the following two kinds of optimization.

3.2.1 Unconstrained optimization

We use a weighted fitness function to balance between objectives and combine them into one fitness function: for each objective, we have a weight for it and we can put more weight on it if we think that one is more important, but the sum of all weights should be equal to 1.

3.2.2 Constrained optimization

For constrained optimization, the problem can be written in the following formula as

$$\begin{cases} \max f(x) = R_S \\ st. \\ g_i(x) \le b_i,\ i = 1,\ldots,m \end{cases} \text{ or } \begin{cases} \min f(x) = C_s \text{ or } O_s \\ \text{st.} \\ \mathrm{R_S} \ge R_0 \\ g_i(x) \le b_i,\ i = 1,\ldots,m \end{cases} \quad (3)$$

Here R_s is system reliability; R_0 is specified minimum system reliability; $g_i(x)$ is the total amount of resource i required; C_s is total system cost; and Q_s is other system measure. These mean that, given required costs or power consumption, the designer tries to find the design with maximal reliability; or given a required reliability, the designer tries to minimize cost or power consumption.

4 NUMERICAL EXAMPLE

To evaluate the performance of the proposed GA method for reliability design system, an example of a cardiac-assist system taken from [4] is implemented. For the block diagram and fault tree model of this system, please refer to [4].

It is tedious to construct the DBN for a system from scratch. Hence, we will draw the DFT for this system first, and then transfer it to DBN. Modeling this system involves dynamic gates: the Backup CPU is a warm spare for the Primary CPU (WSP gate); the two CPUs are dependent on the System Supervisor and the Crossbar Switch. If either of these fails, it will cause both CPUs to fail (PDEP gate). Before building the whole system, we briefly introduce these two gates (WSP & PDEP).

4.1 Warm spare gate

A spare gate models the presence of a set of spare components associated with a main component. When the main component fails, it can be replaced by one of its spares. A failure rate λ and a dormancy factor α ($0 \le \alpha \le 1$) are associated with each spare component. Each spare component can be one of these states: standby with failure rate αλ; working with failure rate λ; or failed. Dependent on α, there are 3 kinds of spare gates: it becomes a Hot Spare Gate (HSP) when α=1; it becomes a Warm Spare Gate (WSP) when $0<\alpha<1$; and it becomes a Cold Spare Gate (CSP) when α=0. Here, let us consider a simple situation where a single component, A, can be substituted by a spare B. They both have 3 options for "CHOICE" or the total redundancy is 3 for "REDUNDANCY". The DFT and DBN are shown in Fig. 2.

We just show the probabilities of failure for node B and node WSP in equation (4).

4.2 Functional Dependency Gate

A PDEP gate models a system where the failure of one component (trigger event) causes some other components (dependent events) to fail with a certain probability. For

example, a failure of A causes both B and C to fail. There are 3 options for each of them, so total redundancy is 3. The DFT and DBN are shown in Fig. 3. The probability of failure for node B(t+1) is shown in equation (5) for CHOICE and (6) for REDUNDANCY, respectively. (Node C(t+1) is the same as node B(t+1), and node FDEP is the same as node A(t+1))

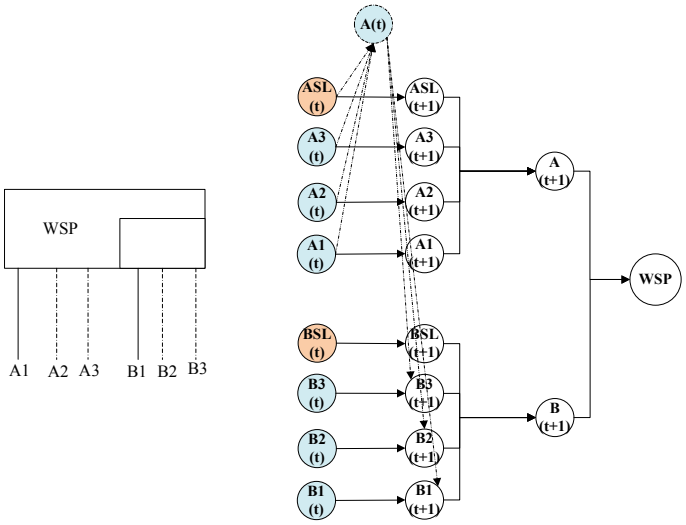

Fig. 2: Fault tree and DBN for WSP gate

$$CHOICE = REDUNDANCY =$$
$$\begin{cases} P(Bi(t+1)=1 \mid Bi(t)=1, A(t)=1) = 1-e^{-\partial\lambda^{Bi}} \\ P(Bi(t+1)=1 \mid Bi(t)=1, A(t)=0) = 1-e^{-\lambda^{Bi}} \\ P(WSP=1 \mid A(t+1)=1) = 1 \\ P(WSP=1 \mid B(t+1)=1) = 1 \\ P(WSP=0 \mid A(t+1)=0, B(t+1)=0) = 1 \end{cases} \quad (4)$$

$$CHOICE \begin{cases} P(B(t+1)=1 \mid Bi(t+1)=1, BSL(t+1)=i, A(t+1)=1) = 1 \\ P(B(t+1)=1 \mid Bi(t+1)=1, BSL(t+1)=i, A(t+1)=0) = p_{FDEP} \\ P(B(t+1)=0 \mid Bi(t+1)=0, BSL(t+1)=i) = 1 \end{cases} \quad (5)$$

4.3 Cardiac assist system

Except for CPU WSP and PDEP gates, all other gates are OR gates. The backup CPU is considered as a WSP to the primary CPU. Both CPUs can fail, but the backup CPU has a reduced failure rate when it is in use. Here the dormancy factor is 0.5. The PDEP gate is used to model the relations between the system supervisor, the crossbar switch, and the two CPUs. The trigger event of the PDEP gate is the OR gate, which has input components of the system supervisor and the crossbar switch; the failures of two CPUs are dependent events of the PDEP gate. For the 7 components in the rectangle, all have 4 potential options, which are labeled as L1, L2, L3 and L4. L4 is always the advanced component with the highest reliability, the lowest power consumption, and the highest cost. In comparison, L1 is the most inferior component, with the lowest reliability, the highest power consumption and the lowest cost. L2 and L3 are in the middle of these two. A redundancy must be assigned to a power supply; the total redundancy level for this component is 4. Based on component choice and the dynamic fault tree, we can build the DBN of the cardiac assist system. Table 1 lists the failure rate, power consumption and cost for all components (values listed here are not exactly the same as those in [4]. Some necessary adjustments are made for illustration purposes).

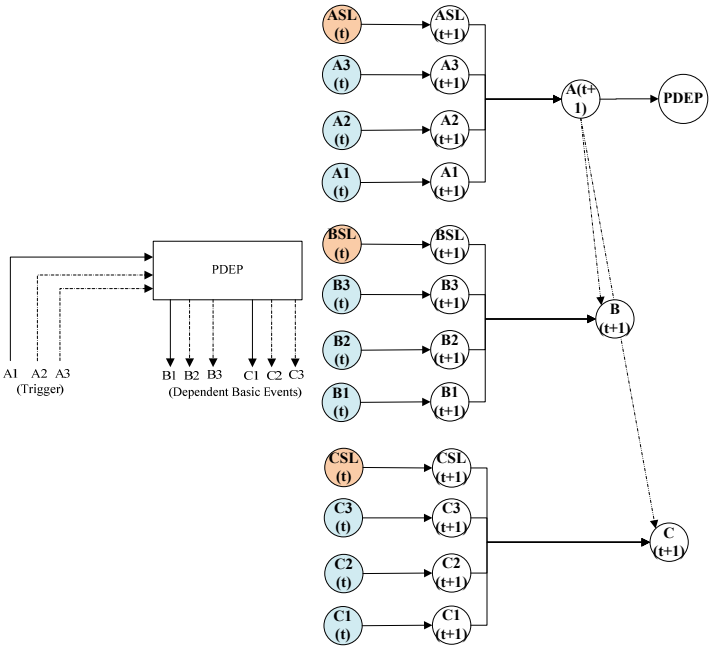

Fig. 3: Fault tree and DBN for PEDP gate

$$REDUNDANCY =$$
$$\begin{cases} P(B(t+1)=1 \mid B1(t+1)=1, BSL(t+1)=1, A(t+1)=1)=1 \\ P(B(t+1)=0 \mid B1(t+1)=0, BSL(t+1)=1)=1 \\ P(B(t+1)=0 \mid B1(t+1)=1, BSL(t+1)=1, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=1 \mid B1(t+1)=1, B2(t+1)=1, BSL(t+1)=2, A(t+1)=1)=1 \\ p(B(t+1)=1 \mid B1(t+1)=1, B2(t+1)=0, BSL(t+1)=2, A(t+1)=1)=1 \\ p(B(t+1)=1 \mid B1(t+1)=0, B2(t+1)=1, BSL(t+1)=2, A(t+1)=1)=1 \\ p(B(t+1)=0 \mid B1(t+1)=0, B2(t+1)=1, BSL(t+1)=2)=1 \\ p(B(t+1)=0 \mid B1(t+1)=1, B2(t+1)=1, BSL(t+1)=2, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=0 \mid B1(t+1)=1, B2(t+1)=0, BSL(t+1)=2, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=0 \mid B1(t+1)=0, B2(t+1)=1, BSL(t+1)=2, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=1 \mid B1(t+1)=1, B2(t+1)=1, B3(t+1)=1, BSL(t+1)=3, A(t+1)=1)=1 \\ p(B(t+1)=1 \mid B1(t+1)=1, B2(t+1)=1, B3(t+1)=0, BSL(t+1)=3, A(t+1)=1)=1 \\ p(B(t+1)=1 \mid B1(t+1)=1, B2(t+1)=0, B3(t+1)=1, BSL(t+1)=3, A(t+1)=1)=1 \\ p(B(t+1)=1 \mid B1(t+1)=0, B2(t+1)=1, B3(t+1)=1, BSL(t+1)=3, A(t+1)=1)=1 \\ p(B(t+1)=1 \mid B1(t+1)=0, B2(t+1)=0, B3(t+1)=1, BSL(t+1)=3, A(t+1)=1)=1 \\ p(B(t+1)=1 \mid B1(t+1)=0, B2(t+1)=1, B3(t+1)=0, BSL(t+1)=3, A(t+1)=1)=1 \\ p(B(t+1)=1 \mid B1(t+1)=1, B2(t+1)=0, B3(t+1)=0, BSL(t+1)=3, A(t+1)=1)=1 \\ p(B(t+1)=0 \mid B1(t+1)=0, B2(t+1)=0, B3(t+1)=0, BSL(t+1)=3)=1 \\ p(B(t+1)=0 \mid B1(t+1)=1, B2(t+1)=1, B3(t+1)=0, BSL(t+1)=3, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=0 \mid B1(t+1)=1, B2(t+1)=0, B3(t+1)=1, BSL(t+1)=3, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=0 \mid B1(t+1)=0, B2(t+1)=1, B3(t+1)=1, BSL(t+1)=3, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=0 \mid B1(t+1)=0, B2(t+1)=0, B3(t+1)=1, BSL(t+1)=3, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=0 \mid B1(t+1)=0, B2(t+1)=1, B3(t+1)=0, BSL(t+1)=3, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=0 \mid B1(t+1)=1, B2(t+1)=0, B3(t+1)=0, BSL(t+1)=3, A(t+1)=0) = p_{FDEP} \\ p(B(t+1)=0 \mid B1(t+1)=1, B2(t+1)=1, B3(t+1)=1, BSL(t+1)=3, A(t+1)=0) = p_{FDEP} \end{cases} \quad (6)$$

Table 1

Component	Failure Rate				Power Consumption				Cost			
	L1	L2	L3	L4	L1	L2	L3	L4	L1	L2	L3	L4
Pump	.0003	.00009	.000002	.000001	14	5	1	.5	80	65	40	30
Motor	.0004	.00002	.000008	.000004	14	7	1	.5	50	40	30	10
Motor Amplifier	.0004	.000028	.000007	.000005	10	5	1	.5	45	35	15	4
Battery	.00015	.00008	.000001	.0000008	15	5	1	.5	35	25	10	5
Primary CPU	.00009	.00006	.00001	.000005	12	8	4	2	35	25	10	5
Backup CPU	.00009	.00006	.00001	.000005	12	6	2	1	35	25	10	5
System Supervisor	.00009	.00002	.00001	.000005	10	5	1	.5	30	25	10	5
Leads	.00001				6				20			
Motor Cable	.000014				2				18			
TEDTS Coil	.000018				3				25			
Power Supply	.00009				2				15			
Crossbar Switch	.000002				2				5			
TEDTS Controller	.00004				4				16			

4.4 Simulation Results

4.4.1 Unconstrained optimization

Here we only consider the tradeoffs between unreliability, design cost and power consumption, so the fitness function is as follows:

$$Fitness(i) = -W_1 \frac{Unreliability(i)}{Max(Unreliability)} - W_2 \frac{Cost(i)}{Max(Cost)} - W_3 \frac{Power(i)}{Max(Power)} \quad (7)$$

$$W_1 + W_2 + W_3 = 1$$

Mission time = 100 hours,
Population = 100,
Maximum generation = 10,
Generation gap = .8,
Probability of crossover = 1,
Probability of mutation = 0.01
Randomly initialize the population,
Binary encoding

The reliability, design cost and power consumption for each design are listed in Table 2 after optimization.

Table 2

Design Items	W1=1 W2=0 W3=0	W1=0 W2=1 W3=0	W1=0 W2=0 W3=1	W1=.5 W2=.2 W3=.3	W1=.1 W2=.4 W3=.5
Unreliability	0.01171	0.14209	0.02057	0.0125	0.02332
Cost	454	163	409	369	334
Power Used	30.5	106	24.5	29.5	31.5
Best Solution	[4 4 4 4 4 4 4 4]	[1 1 1 1 1 1 1 1]	[4 4 4 4 4 4 4 1]	[3 3 3 3 4 3 4 2]	[3 3 1 3 4 3 4 1]

For design items [W1=1, W2=0, W3=0], all L4 components and redundancy 4 are chosen as an optimal solution, which gives maximal reliability; for design items [W1=0, W2=1, W3=0], all L1, the cheapest component, and redundancy 1 are chosen, which can give out the minimal cost; for design items [W1=0, W2=0, W3=1], all L4 components are chosen again, but redundancy 2 is chosen to get minimal power consumption. All of these optimal solutions are expected.

Other cases are analyzed by tuning W1, W2 and W3. Generally the designer tries various W1, W2 and W3 and then compares the reliability, cost and power consumption results to find the optimal design.

4.4.2 Constrained optimization

The formulations for constrained optimization are as in equation (8).

$$\begin{cases} \max\ fitness = -K_1 Unreliability - K_2 Cost - K_3 Power \\ st \\ (1-K_1) Unreliability \le Unreliability_{required} \\ (1-K_2) Cost \le Cost_{required} \\ (1-K_3) Power \le Power_{required} \\ K_1 + K_2 + K_3 = 1 \end{cases} \quad (8)$$

The equations indicate that given the required cost or power consumption (cost or power consumption cannot exceed a level), the designer tries to find the optimal design with maximal reliability; or given a required reliability (reliability must exceed a level), the designer tries to find the optimal design with minimal cost or power consumption. Here K1, K2 and K3 are binary numbers.

All the GA settings are the same as the unconstrained optimization. The reliability, design cost and power consumption for each design are listed in Table 3 as in the following.

Table 3

Design Items	K1=1 K2=0 K3=0 Cost <=200 Power <=100	K1=0 K2=1 K3=0 Unreliability <=0.015 Power <=55	K1=0 K2=0 K3=1 Unreliability <=0.02 Cost <=300
Unreliability	0.06859	0.01463	0.01723
Cost	199	299	294
Power Used	89	53.5	50
Best Solution	[1 2 2 1 1 1 1 1]	[3 3 2 3 1 1 4 2]	[3 2 2 3 3 2 2 2]

For the design item [K1=1, K2=0, K3=0], the designer wants to find the most reliability design with maximal budget cost 200 and maximal power consumption requirement 100. The proposed optimization framework gives out a solution with unreliability 0.06859, cost 199 and power consumption 89. For the design item [K1=0, K2=1, K3=0], the designer wants to find the minimal cost design with reliability greater than 0.985 and power consumption less than 55. However, the optimization framework gives out a solution with unreliability 0.01463, cost 299 and power consumption 53.5. The last design item means the designer wants to find the minimal power consumption design with minimal reliability 0.98 and maximal cost 300; the cost of the solution is 294 with unreliability 0.01723 and power consumption 50. All the results above show that the optimization framework works well and can aid designers in finding the desired optimal design. Other cases are analyzed by choosing different combinations of Ks and constraints.

5 CONCLUSION

In this paper, a framework based on Dynamic Bayesian Networks and the Genetic Algorithm has been proposed for solving the redundancy allocation and component selection problem in the reliable system design process. The reliability design problems are usually formulated based on simple system structure (series, parallel or k-out-of-n) without component dependency; here we extend it to a more complicated system with dynamic behaviors. The Dynamic Bayesian Networks, which are not only capable of modeling complex component behaviors and interactions, but also alleviate the problem encountered (e.g. state space explosion) by other reliability modeling methods, are used to estimate the system reliability. The Genetic Algorithm, a derivative-free heuristic method, is used to solve the design optimization problem with respect to different objectives (e.g. reliability, cost and so on) to give out the optimal design for designers. The proposed framework is applied to a classical example from literature and the results show that it is a very powerful and flexible tool for system design.

REFERENCES

1. J. Holland, *Adaptation in Natural and Artificial Systems*,University of Michigan Press, 1975
2. D. W. Coit, A. E. Smith, Reliability Optimization of Series-Parallel Systems using a Genetic Algorithm, *IEEE Transaction on Reliability*, 45(2), 1996, 254-266
3. D. W. Coit, Cold-standby redundancy optimization for nonrepairable system. *IIE Transaction*,33(6), 2001, 471-478
4. Y. Ren, J. B. Dugan, Design of Reliable System Using Static & Dynamic Fault Trees, *IEEE Transactions on Reliability*, 47(3), 1998, 234-244
5. A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla, Improving the Analysis of Dependable Systems by Mapping Fault Trees into Bayesian Networks, *Reliability Engineering and System Safety*, 71, 2001, 249-260
6. P. Weber, L. Jouffe, Complex system reliability modeling with Dynamic Object Oriented Bayesian Networks (DOOBN). *Reliability Engineering & System Safety*, 91(2):149–162, 2006
7. K., Murphy, *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, UC Berkley, 2002.

BIOGRAPHIES

Dingzhou Cao, Ph.D.
Research Scientist
ReliaSoft Corporation
1450 S. Eastside Loop
Tucson, Arizona 85710, USA

Email: Dingzhou.Cao@ReliaSoft.com

Dr. Dingzhou Cao is a Research Scientist at ReliaSoft Corporation. He received his Ph.D. in Industrial Engineering from Wayne State University.

Shaobai Kan, Ph.D.
Assistant Professor
John Jay College of Criminal Justice, CUNY
Room 4226N, 445 W. 59th Street
New York, NY 10019

Email: skan@jjay.cuny.edu

Dr. Shaobai Kan is an Assistant Professor in the Department of Mathematics and Computer Science at John Jay College of Criminal Justice, CUNY. He received his Ph.D. in Applied Mathematics from Wayne State University.

Yu Sun, Ph.D.
Ph.D. Candidate
Wayne State University
FAB 1118, 656 W. Kirby
Detroit, MI 48202, USA

Email: ysun@wayne.edu

Dr. Yu Sun is a Ph.D. candidate in the Department of Mathematics at Wayne State University. She received her Ph.D. in Management Engineering from Donghua University, Shanghai, China.