

Redundancy allocation for multi-state systems using physical programming and genetic algorithms

Zhigang Tian, Ming J. Zuo*

Department of Mechanical Engineering, University of Alberta, Edmonton, Alta., Canada T6G 2G8

Available online 9 January 2006

Abstract

This paper proposes a multi-objective optimization model for redundancy allocation for multi-state series–parallel systems. This model seeks to maximize system performance utility while minimizing system cost and system weight simultaneously. We use physical programming as an effective approach to optimize the system structure within this multi-objective optimization framework. The physical programming approach offers a flexible and effective way to address the conflicting nature of these different objectives. Genetic algorithm (GA) is used to solve the proposed physical programming-based optimization model due to the following three reasons: (1) the design variables, the number of components of each subsystems, are integer variables; (2) the objective functions in the physical programming-based optimization model do not have nice mathematical properties, and thus traditional optimization approaches are not suitable in this case; (3) GA has good global optimization performance. An example is used to illustrate the flexibility and effectiveness of the proposed physical programming approach over the single-objective method and the fuzzy optimization method.

© 2005 Published by Elsevier Ltd.

Keywords: Multi-state series–parallel system; Multi-objective optimization; Genetic algorithm; Physical programming; Fuzzy optimization

1. Introduction

Traditional reliability theory assumes that a system and its components may take only two possible states, working or failed. However, many practical systems can actually perform their intended functions at more than two different levels, from perfectly working to completely failed. These kinds of systems are known as multi-state systems [1]. A multi-state system reliability model provides more flexibility for modeling of system conditions than a binary reliability model. Many performance measures of multi-state systems have been proposed, and one of the most popular performance measures is system performance utility [2].

In practical situations involving reliability optimization, there often exist mutually conflicting goals such as maximizing system utility and minimizing system cost and weight [3]. Current multi-state optimization models

treat one goal as the objective function of the optimization model and the other goals as constraints. Levitin et al. [4] studied the redundancy optimization problem for series–parallel multi-state systems, and the model they used aimed at minimizing the system cost under the constraints of system availability. Levitin and Lisnianski [5] presented a structure optimization approach for multi-state systems with two failure modes. They proposed two optimization problem formulations: one aims at maximizing system availability and the other aims at providing the maximal proximity of the expected system performance to the desired levels for both the open mode and the close mode, while satisfying the system cost and reliability requirements. Liu et al. [6] presented a neural network approximation approach for optimal design of continuous-state parallel–series systems, so as to reduce the computational complexity in the utility evaluation of continuous-state systems. Their model was built to maximize system utility subject to system cost constraints.

However, it is very difficult to specify in advance the constraint values for the goals used as constraints. After a solution is obtained, we often need to modify these

Abbreviations: DM, decision maker; OVO, One vs. Others

*Corresponding author. Tel.: +1 780 492 4466; fax: +1 780 492 2200.

E-mail address: ming.zuo@ualberta.ca (M.J. Zuo).

Nomenclature			
\bar{g}_i	class function of objective i	U	system utility
g_{ij}	boundary value of preference ranges for objective $ij = 1, 2, \dots, 5$	C	system cost
N	number of subsystems (stage)	W	system weight
S_i	subsystem (stage) $i, 1 \leq i \leq N$	s	state s of component or system
n_i	number of components in S_i	u_s	the utility when system is in state s
h_i	component version in S_i	c_i	cost function of component in S_i
H_i	the number of types of components available for S_i	w_i	weight function of component in S_i
M	the maximum state level of the components and the system	\mathbf{n}	(n_1, n_2, \dots, n_N)
x_{ij}	the state of component j of subsystem i	\mathbf{h}	(h_1, h_2, \dots, h_N)
p_{ik}	probability of a component of subsystem i in state k	\bar{g}_U	class function of system utility
\mathbf{x}	a vector representing the states of all components in the multi-state system	\bar{g}_C	class function of system cost
$\phi(\mathbf{x})$	state of the system, $\phi(\mathbf{x}) = 0, 1, \dots, M$	\bar{g}_W	class function of system weight
		U_0	system utility constraint value
		C_0	system cost constraint value
		W_0	system weight constraint value
		$\mu_{\bar{U}}$	fuzzy membership function of system utility
		$\mu_{\bar{C}}$	fuzzy membership function of system cost
		$\mu_{\bar{W}}$	fuzzy membership function of system weight

constraint values to find a better tradeoff between goals such as system utility and system cost. But finding the most appropriate constraint values is a trial and error process, it is time-consuming, and there is no clear guidance as to how to converge to the right set of constraint values. Particularly, when there are many constraint values that need to be specified, it is almost impossible to find the most appropriate values for these constraints.

There has been intensive research on redundancy allocation of binary systems considering multiple objectives. Sakawa [7] presented a multi-objective formulation to maximize reliability and minimize cost for system structure optimization by using the surrogate worth trade-off method. Inagaki et al. [8] proposed to maximize reliability and minimize cost and weight by using an interactive optimization method. Park [9] used fuzzy logic theory to analyze a multi-objective reliability apportionment problem for a two-component series system. Dhingra [10] and Rao and Dhingra [11] studied the reliability and redundancy allocation problem for a four-stage and a five-stage over-speed protection system, using crisp and fuzzy multi-objective optimization approaches, respectively. Huang [12] proposed an approach for fuzzy multi-objective optimization decision-making involving a series reliability system. Ravi et al. [13] modeled the problem of optimizing the reliability of a complex system as a fuzzy multi-objective optimization problem. Huang et al. [14] applied a proposed interactive physical programming approach to reliability-redundancy optimization of a binary-state system. However, this approach is too complicated to be used conveniently in practical problems, and the improvement on the solution's quality is trivial compared to using the original physical programming approach [15].

In this paper, we present a multi-objective optimization model for redundancy allocation for multi-state series-par-

allel systems, which optimizes the goals of system utility, system cost and system weight simultaneously. The physical programming method is used as an effective approach to optimize the system structure within this model's framework. Physical Programming eliminates the typical iterative process involving the adjustment of the physically meaningless weights, which is required by virtually all other multi-objective optimization methods, and thus substantially reduces the computational intensities. The DMs' preferences are specified individually on each goal through physically meaningful values, which makes the physical programming method easy to use and advantageous in dealing with a large number of objectives. Physical programming has proved its effectiveness in addressing a wide array of multi-objective problem [15–19]. In the present application, we apply physical programming to the redundancy allocation for multi-state systems. As will be shown, physical programming offers a flexible and effective approach for the optimal design of multi-state system.

Genetic algorithm (GA) is a very powerful optimization approach [20], and it is used to solve the proposed physical programming-based optimization model due to the following three reasons: (1) the design variables, the number of components of each subsystems, are integer variables; (2) the objective functions in the physical programming-based optimization model do not have nice mathematical properties, and thus traditional optimization approaches are not suitable in this case; (3) GA has good global optimization performance. An example is used to illustrate the flexibility and effectiveness of the proposed physical programming approach over the approaches that are traditionally used in binary and multi-state redundancy allocation problems, e.g., the single-objective method and the fuzzy optimization method.

2. Physical programming synopsis

Physical programming [15] is a multi-objective optimization tool that explicitly incorporates the DM’s preferences on each design metric into the optimization process. Within the physical programming procedure, the design metrics are classified into four classes: smaller is better (i.e., minimization), larger is better (i.e., maximization), value is better, and range is better. There are two so-called class functions, one soft and one hard, with respect to each class [15]. The hard class functions are used to represent the constraints, while the soft class functions become additive constituent components of the aggregate objective function (to be minimized) of the optimization model. Consider for example the case of class-1 soft class function (class 1-S), the qualitative meaning of the preference function is depicted in Fig. 1. The value of the design metric, g_i , is on the horizontal axis, and the corresponding class function, \bar{g}_i , is on the vertical axis. A lower value of the preference function is better than a higher value thereof.

Physical programming allows the DM to express ranges of differing levels of preference with respect to each design metric with more flexibility and specificity than by simply declaring minimize, maximize or equal to. For Class 1-S, shown in Fig. 1, the ranges are defined as follows:

- highly desirable range: $g_i \leq g_{i1}$;
- desirable range: $g_{i1} \leq g_i \leq g_{i2}$;
- tolerable range: $g_{i2} \leq g_i \leq g_{i3}$;
- undesirable range: $g_{i3} \leq g_i \leq g_{i4}$;
- highly undesirable range: $g_{i4} \leq g_i \leq g_{i5}$;
- unacceptable range: $g_i \geq g_{i5}$.

The parameters g_{i1} through g_{i5} are physically meaningful constants associated with each design metric i . What the DM needs to do in the physical programming framework is just to specify the values of the parameters g_{i1} , g_{i2} , g_{i3} , g_{i4} , and g_{i5} for each design metric i , and the class function can be completely determined by these parameters. Refer to Messac [15] for the details on how to built class functions from these boundary parameters.

The range limits define the intra-criteria preference, while the “One vs. Others” criteria rule (OVO rule)

describe the inter-criteria preference. Suppose there are two options: (1) full reduction for *one* criterion across a given preference range, say, the tolerable range; (2) full reduction for all the other criteria across the next better range, say, the desirable range. The OVO rule decides that option (1) is preferred over option (2). For example, assume that we have four criteria to be minimized, criterion 1–4. We say that the reduction of criterion 1 from the right boundary to the left boundary of the tolerable range is preferred over the reductions of criterion 2, 3, and 4 all from the right boundary to the left boundary of the desirable ranges. The OVO rule is built into the generated class function of each criterion.

Class functions also transform design metrics with disparate units and physical meaning into a dimensionless scale. The aggregate objective function is built by combining all the soft class functions. In redundancy allocation problems, there are only Class 1-S functions (to be minimized) and Class 2-S functions (to be maximized), thus the physical programming problem model has the following form [15]:

$$\begin{aligned} \min_x g(\mathbf{x}) &= \log_{10} \left\{ \frac{1}{n_{sc}} \sum_{i=1}^{n_{sc}} \bar{g}_i[g_i(\mathbf{x})] \right\}, \\ \text{s.t. } g_i(\mathbf{x}) &\leq g_{i5} \quad (\text{for class 1-S}), \\ g_i(\mathbf{x}) &\geq g_{i5} \quad (\text{for class 1-S}), \\ x_{jm} &\leq x_j \leq x_{jM}, \end{aligned} \tag{1}$$

where x_{jm} and x_{jM} represent corresponding minimum and maximum values, n_{sc} is the number of the soft design metrics that the problem comprises.

The physical programming approach has the following characteristics: (1) it eliminates the iterative weight-adjusting process, thus substantially reduces the computational intensity; (2) the DM only needs to specify desirability ranges for each design metric, not those meaningless weights, which makes this approach very friendly to users; (3) the DM’s preferences are specified on each design metric individually, therefore, physical programming is suitable to deal with a larger number of design objectives. While in the weight-based methods, there is no reasonable way to specify the weights for the objectives in advance.

3. Physical programming-based multi-state system optimization model and solution approach

In this section, we will present the approach to evaluate the system utility of a multi-state series-parallel system, formulate the multi-state redundancy allocation problem model based on physical programming, and give the solution approach using GA for the physical programming-based optimization model.

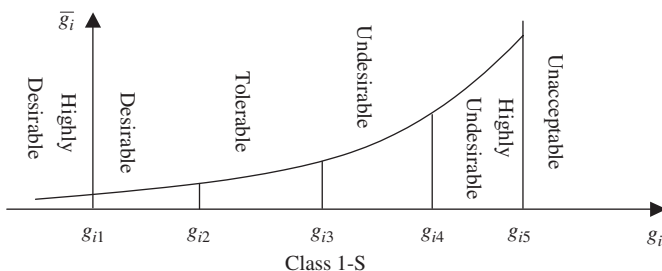


Fig. 1. Qualitative meaning of soft class function.

3.1. System utility evaluation of multi-state series–parallel system

As shown in Fig. 2, a multi-state series–parallel system consists of N subsystems, S_1 to S_N , connected in series. Each subsystem, say S_i , has n_i identical components connected in parallel.

To design such a system, we need to select the type of components to use for each subsystem and determine the number of components of the selected type [4]. The following assumptions are used:

- (1) For each subsystem S_i , there are H_i types of components available in the market. For a type $h_i(1 \leq h_i \leq H_i)$ component, its state probability distribution, cost and weight are specified.
- (2) The states of the components in a subsystem are identically and independently distributed.
- (3) The component and the system may be in $M + 1$ possible states, namely, $0, 1, 2, \dots, M$.

According to multi-state system definition of Barlow and Wu [21], the state of a parallel system is defined to be the state of the best component in the system, and the state of a series system is defined to be the state of the worst component in the system. Hence, the system state of the series–parallel shown in Fig. 2 is

$$\phi(x) = \min_{1 \leq i \leq N} \max_{1 \leq j \leq n_i} x_{ij}. \tag{2}$$

And the probability that the system is in state s or above ($s = 0, 1, \dots, M$) is

$$Pr(\phi(x) \geq s) = \prod_{i=1}^N \left[1 - \left(1 - \sum_{k=s}^M p_{ik} \right)^{n_i} \right], \tag{3}$$

where $p_{ik} = Pr(x_{ij} = k)$ for any j .

The index of system utility is used to measure the performance of a multi-state series–parallel system [2]

$$U = \sum_{s=0}^M u_s \cdot Pr(\phi(x) = s), \tag{4}$$

where U is the expected system utility, and u_s is the utility when system is in state s .

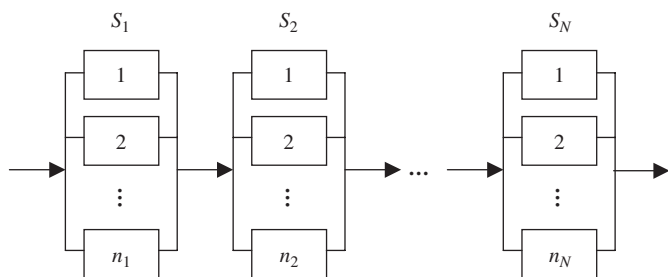


Fig. 2. Structure of a multi-state series–parallel system.

For given h_1, h_2, \dots, h_N , and n_1, n_2, \dots, n_N values, the total cost of the system is expressed as [10]

$$C = \sum_{i=1}^N c_i(h_i)[n_i + \exp(n_i/4)]. \tag{5}$$

The system weight is [10]

$$W = \sum_{i=1}^N w_i(h_i)n_i \exp(n_i/4). \tag{6}$$

3.2. Physical programming-based redundancy allocation model

In the redundancy allocation problem for multi-state series–parallel system, the design variables are the component version vector \mathbf{h} and component number vector \mathbf{n}

$$\mathbf{h} = (h_1, h_2, \dots, h_N), \quad \mathbf{n} = (n_1, n_2, \dots, n_N). \tag{7}$$

The design metrics under consideration are system utility, system cost and system weight. The system utility, which is to be maximized, is described using class-2S class function in the physical programming approach framework. The system cost and system weight, which are to be minimized, are both described using class-1S class function.

The multi-objective optimization model of the multi-state system is formulated as

$$\begin{aligned} \min_{\mathbf{n}, \mathbf{h}} g(\mathbf{n}, \mathbf{h}) &= \log_{10} \left\{ \frac{1}{3} [\bar{g}_U(U(\mathbf{n}, \mathbf{h})) + \bar{g}_C(C(\mathbf{n}, \mathbf{h})) \right. \\ &\quad \left. + \bar{g}_W(W(\mathbf{n}, \mathbf{h}))] \right\} \\ \text{s.t. } U(\mathbf{n}, \mathbf{h}) &\geq U_0, \\ C(\mathbf{n}, \mathbf{h}) &\leq C_0, \\ W(\mathbf{n}, \mathbf{h}) &\leq W_0, \\ 0 < h_i &\leq H_i, \quad i = 1, 2, \dots, N \\ 0 < n_i &, \quad i = 1, 2, \dots, N \\ n_i, h_i &\text{ are integers,} \end{aligned} \tag{8}$$

where $g(\mathbf{n}, \mathbf{h})$ is the aggregate objective function, \bar{g}_U, \bar{g}_C and \bar{g}_W are the class functions of system utility, system cost and system weight, respectively, H_i is the number of types for component C_i , and U_0, C_0 and W_0 are the constraint values, which are equal to the boundaries of the acceptable ranges of the corresponding design metrics.

3.3. GA as a solution approach

GA is a very powerful optimization approach with two key advantages. (1) Flexibility in modeling the problem. GA has no strict mathematical requirements, such as derivative requirement, on the objective functions and constraints. The only requirement is that the objective functions and constraints can be evaluated in some way. GA is also suitable for dealing with those problems

including discrete design variables. (2) Global optimization ability. GA has been recognized as one of the most effective approaches in searching for the global optimal solution.

The procedure of GA is as follows:

- (1) *Initialization*: Set the size of population and the length of the chromosome. Set $k = 0$, and generate the initial population $P(0)$.
- (2) *Evaluation*: Calculate the fitness value of each chromosome of the current population $P(k)$. Save the chromosome $B(k)$ with the best fitness value.
- (3) *Select*: Select chromosomes from the current population based on their fitness values to form a new population $P(k+1)$.
- (4) *Cross*: One point crossover is used to $P(k+1)$.
- (5) *Mutate*: Implement even mutation on $P(k+1)$.
- (6) *Duplication*: Use $B(k)$ to replace the first chromosome in $P(k+1)$.
- (7) If the maximal iteration is reached, terminate the procedure and output the result. Otherwise, set $k = k + 1$, and go to step (2).

3.4. Strength of the proposed physical programming-based multi-state system optimization approach

The redundancy allocation problem of multi-state system is formulated as multi-objective optimization problem in this paper, because there are multiple objectives such as utility, cost and weight under consideration. The proposed multi-objective optimization model provides us a systematic framework to deal with these multiple objectives. Previous studies used single-objective optimization methods to solve such problems, with one objective as the optimization criterion and the other objectives as constraints. The single-objective optimization method is easy to understand, but it is difficult to specify these constraint values in advance. After a solution is obtained, it is often needed to modify these constraint values to find a better tradeoff between these goals. But finding the most appropriate constraint values is a trial and error process, it is time-consuming, and there is no guarantee that a satisfactory solution will be obtained in the end.

Multi-objective optimization methods have been applied to redundancy optimization of binary-state systems, and a popular one among these methods is fuzzy optimization method [10,11,13]. Fuzzy optimization method allows DM to express his or her preferences on different objectives using fuzzy membership functions. However, fuzzy membership functions are not flexible enough to clearly express DM's preferences, and there is no internal mechanism to control the tradeoff among different objectives.

Physical programming, however, provides more flexibility for the DM to specify his or her preferences on each objective using class functions. The DM can specify different levels of preference on each objective easily. In

addition, the OVO rule of physical programming provides the internal mechanism to control tradeoff among different objectives. In addition to optimizing each individual objective, physical programming will drive the optimal values of different objectives to preference ranges close to one another.

4. Example

An example is used to illustrate the redundancy allocation procedure for a multi-state series–parallel system using physical programming-based multi-objective optimization approach, and demonstrate the advantages of the physical programming approach over single-objective method and the fuzzy optimization method.

A multi-state series–parallel system with four subsystems is considered. The state space of the component and the system is $\{0, 1, 2, 3\}$. Suppose we have different versions of components for each subsystem, with their characteristics listed in Table 1, which gives component cost $c_i(h_i)$, component weight $w_i(h_i)$, and the state distribution for each component, $p_{i0}–p_{i3}$. The system utility u_s with respect to the corresponding system state s is shown in Table 2.

Three methods, single-objective optimization method, fuzzy optimization method and the physical programming approach, are used to optimize this multi-state system so as

Table 1
Characteristics of available components

S_i	h_i	p_{i0}	p_{i1}	p_{i2}	p_{i3}	$c_i(h_i)$	$w_i(h_i)$
1	1	0.100	0.450	0.250	0.200	0.545	7
	2	0.060	0.450	0.200	0.290	0.826	3
	3	0.045	0.400	0.150	0.405	0.975	10
	4	0.038	0.350	0.160	0.452	1.080	8
2	1	0.050	0.450	0.300	0.200	0.550	12
	2	0.040	0.400	0.300	0.260	0.630	5
	3	0.040	0.300	0.320	0.340	0.740	8
	4	0.030	0.250	0.250	0.470	0.900	10
	5	0.025	0.215	0.180	0.580	1.150	12
3	1	0.145	0.625	0.130	0.100	0.250	10
	2	0.110	0.400	0.250	0.240	0.380	12
	3	0.065	0.450	0.230	0.255	0.494	13
	4	0.080	0.300	0.300	0.320	0.625	15
	5	0.050	0.250	0.250	0.450	0.790	13
	6	0.038	0.235	0.240	0.487	0.875	17
4	1	0.115	0.535	0.200	0.150	0.545	10
	2	0.074	0.550	0.186	0.190	0.620	15
	3	0.045	0.440	0.215	0.300	0.780	12
	4	0.035	0.330	0.250	0.385	1.120	14

Table 2
System utility with respect to each system state

s	0	1	2	3
u_s	0.0	0.5	0.8	1.0

to maximize system performance utility and minimize system cost and system weight simultaneously. Comparative studies are made to show the advantages of the physical programming approach over the other two methods.

GA is used to solve the formulated optimization models for all the three methods. The GA parameter settings we used in this example are as follows. The decimal encoding is used. The population size is chosen to be 100. The chromosome length is 12: the first four digits are used to represent the component version variables h_1-h_4 , and the rest eight digits are used to represent the numbers of components n_1-n_4 , each with two digits. We use the roulette-wheel selection scheme, one-point cross operator with cross rate of 0.25, and even mutation operator with mutate rate of 0.1.

In any iteration of GA, we have a population of 100 candidate solutions to be evaluated. Suppose the single aggregate objective function to be minimized is f . f^{\max} and f^{\min} are the maximum and minimum value of the population in current iteration. The fitness value for a candidate solution with aggregate objective function value f is

$$F = \frac{f^{\max} - f + 0.3(f^{\max} - f^{\min})}{(1 + 0.3)(f^{\max} - f^{\min})},$$

where the number 0.3 is the so-called “selection pressure” [22]. The “selection pressure” factor in the formula above is used to adjust the fitness value of a candidate solution based on its aggregate objective function value. When the “selection pressure” is bigger, the probability for those candidate solutions with relatively large aggregate objective function values will become higher, while that for those with relatively small aggregate objective function values will become lower. If the “selection pressure” is set to be zero, the probability for the candidate solution with the largest aggregate objective function value will be zero.

4.1. Single-objective optimization method

First, we use single-optimization method to get the optimal redundancy allocation scheme. Here, system utility is used as the objective, and system cost and system weight are used as constraints in the optimization model:

$$\begin{aligned} & \max_{n,h} U(\mathbf{n}, \mathbf{h}) \\ & \text{s.t. } C(\mathbf{n}, \mathbf{h}) \leq C_0, \\ & \quad W(\mathbf{n}, \mathbf{h}) \leq W_0, \\ & \quad 0 < h_i \leq H_i, \quad i = 1, 2, \dots, N, \\ & \quad 0 < n_i, \quad i = 1, 2, \dots, N, \\ & \quad n_i, h_i \text{ are integers,} \end{aligned} \tag{9}$$

where the utility constraint value and the weight constraint value are chosen as $C_0 = 45$ and $W_0 = 1000$.

Table 3
Optimization results using different methods

	Single-objective optimization	Fuzzy optimization	Physical programming
U	0.9654	0.9492	0.9245
C	38.7021	32.2833	27.9569
W	985.8467	699.2584	548.8717
\mathbf{h}	(4, 5, 6, 4)	(4, 5, 6, 4)	(4, 5, 5, 4)
\mathbf{n}	(6, 5, 4, 6)	(5, 4, 4, 5)	(4, 3, 4, 5)

Solve the constrained single-objective optimization problem formulated in (9), and we get the optimization results shown in Table 3. After investigating the optimization result, the DM may think that the utility goal is better than enough, and he wants to improve the cost and weight goals by partly sacrificing the utility goal. But it is hard to specify the new constraint values for cost and weight design metrics. The constraint-adjusting process is a trial-and-error process, and it is quite time-consuming.

4.2. Fuzzy optimization method

Like Dhingra’s work [10], we use the logarithm sigmoid function to define the fuzzy membership functions for the design metrics of system utility, cost and weight, since the logarithm sigmoid function is a popular nonlinear function to define fuzzy membership functions. The standard logarithm sigmoid function is

$$f(x) = 1/(1 + e^{-x}). \tag{10}$$

We have $f(5) = 0.9933$, $f(-5) = 0.0067$. Hence, we use the range $[-5, 5]$ of the standard logarithm sigmoid function to define the fuzzy membership functions for the three design metrics (shown in Fig. 3):

$$\mu_{\tilde{U}}(x) = \begin{cases} 0, & x < 0.9, \\ \frac{f[100(x-0.9)-5]-f(-5)}{f(5)-f(-5)}, & 0.9 \leq x \leq 1, \end{cases} \tag{11}$$

$$\mu_{\tilde{C}}(x) = \begin{cases} 0, & x > 45, \\ \frac{f[(x-25)/2-5]-f(5)}{f(-5)-f(5)}, & 25 < x \leq 45, \\ 1, & x \leq 25, \end{cases} \tag{12}$$

$$\mu_{\tilde{W}}(x) = \begin{cases} 0, & x > 1000, \\ \frac{f[(x-400)/40-5]-f(5)}{f(-5)-f(5)}, & 400 < x \leq 1000, \\ 1, & x \leq 1000. \end{cases} \tag{13}$$

There are two critical values for each fuzzy membership function. For instance, for the cost objective, the two critical values are 25 and 45, which are the starting point and the ending point of the sigmoid part of the membership function, respectively. To ensure a fair comparison, for the cost and weight objectives, we select the critical points with respect to membership function value 0 as the constraint values used in the single-optimization method. The critical

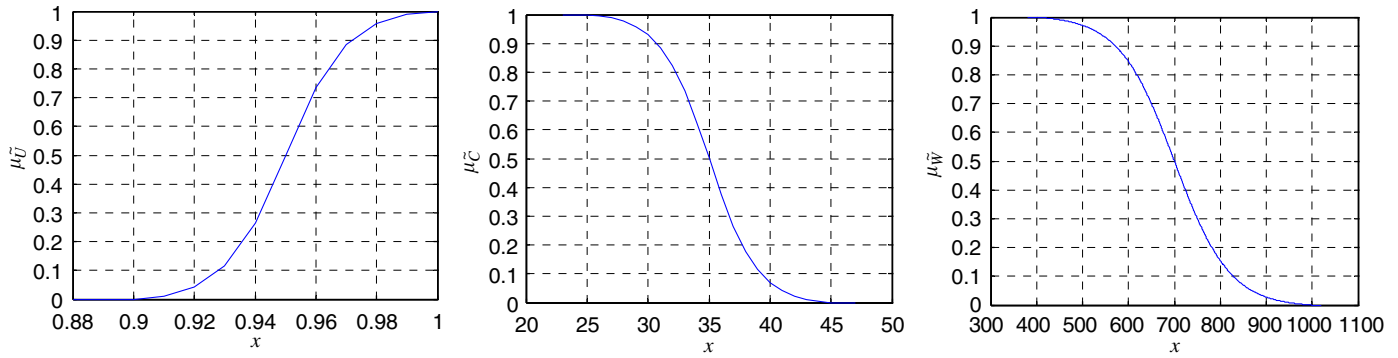


Fig. 3. Fuzzy membership functions.

point for the utility objective with respect to membership function value 0 is set to be 0.9.

The aggregate objective function is defined as

$$\mu_{\tilde{D}} = \min(\mu_{\tilde{U}}(U), \mu_{\tilde{C}}(C), \mu_{\tilde{W}}(U)), \tag{14}$$

where the fuzzy set \tilde{D} is the intersection set of fuzzy sets \tilde{U} , \tilde{C} and \tilde{W} . And the fuzzy multi-objective optimization model is formulated as

$$\begin{aligned} &\max \mu_{\tilde{D}}, \\ &\text{s.t. } U(\mathbf{n}, \mathbf{h}) \geq U_0, \\ &C(\mathbf{n}, \mathbf{h}) \leq C_0, \\ &W(\mathbf{n}, \mathbf{h}) \leq W_0, \\ &0 < h_i \leq H_i, \quad i = 1, 2, \dots, N, \\ &0 < r_i, \quad i = 1, 2, \dots, N, \\ &r_i, h_i \text{ are integers,} \end{aligned} \tag{15}$$

where the constraint values U_0 , C_0 and W_0 are equal to the critical points for the utility objective with respect to membership function value 0.

Solving the optimization above, we get the optimization results shown in Table 3. Compared to the results obtained using single-objective optimization method, the cost and weight objectives are improved, while the utility objective deteriorates. The fuzzy optimization method aims at optimizing the three objectives simultaneously, and finding a compromising solution.

4.3. Physical programming approach

In the physical programming framework, the utility objective has Class-2S class function (larger is better), while the cost and weight objectives have Class-1S class functions (smaller is better). The class functions settings for the three objectives are shown in Table 4. In order to get a result with better system cost performance, we enforce more strict requirements on the cost objective through the class function setting.

For the purpose of a fair comparison, the constraint values in the physical programming optimization model formulated in (8) are set to be the same as those in fuzzy

Table 4
Physical programming class functions setting

	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}
Utility	0.99	0.98	0.95	0.92	0.9
Cost	15	20	25	30	45
Weight	400	500	600	800	1000

optimization models, and they are also equal to the g_{i5} values for the three objectives.

Solving the physical programming optimization model, we get the optimization results shown in Table 3. The cost objective is improved greatly. The optimal utility and cost values both fall into the undesirable ranges, while the optimal weight value falls into the tolerable range.

From the optimization procedures and results, we can find that the physical programming method has two advantages over the fuzzy optimization method:

- (1) The class function of physical programming provides more flexibility for the DM to specify his or her preferences on each objective. The DM can specify different levels of preference on each objective. By modifying the preference ranges of the cost objective, we can get a greatly improved optimal cost value. Using fuzzy optimization, however, the DM can only specify two critical points for each objective. There are different kinds of fuzzy membership functions available, e.g. sigmoid membership function and linear membership function, but it would not provide more flexibility, either.
- (2) The class function provides the preference inside a single objective, and the OVO rule of physical programming provides the preferences among different objectives. Therefore, besides optimizing each individual objective, physical programming will drive the optimal values of different objectives to preference ranges close to one another. The class functions and the OVO rule lead to optimal solution that best satisfies the DM's preferences on different objectives.
- (3) A problem of using fuzzy optimization approach is that maybe the resulted optimal cost and weight are good

while the utility objective is totally unsatisfying. Of course we can adjust the fuzzy membership functions when encountering this problem, but we cannot ensure that the optimal results will be satisfying next time. When using the physical programming approach, however, with class functions accurately describing the DM's preferences on each objective and with the OVO rule as the inter-criteria preference, we can ensure to get satisfying optimal results with respect to each design criterion.

5. Conclusions

This paper proposes a multi-objective optimization model for redundancy allocation for multi-state series–parallel systems. This model seeks to maximize system performance utility while minimizing system cost and system weight simultaneously. Physical programming is used as an effective approach to optimize the system structure within this multi-objective optimization framework. The physical programming approach offers a flexible and effective way to address the conflicting nature of these different objectives. GA is used in solving the proposed physical programming-based optimization model. The example illustrates the flexibility and effectiveness of the proposed physical programming approach over the single-objective method and the fuzzy optimization method.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Alberta Ingenuity Fund.

References

- [1] Kuo W, Zuo MJ. Optimal reliability modeling: principles and applications. Hoboken, NJ: Wiley; 2003.
- [2] Aven T. On performance-measures for multistate monotone systems. Reliab Eng Syst Saf 1993;41(3):259–66.
- [3] Kuo W, Prasad VR, Tillman FA, Hwang CL. Optimal reliability design: fundamentals and applications. Cambridge: Cambridge University Press; 2001.
- [4] Levitin G, Lisnianski A, Ben-Haim H, et al. Redundancy optimization for series–parallel multi state systems. IEEE Trans Reliab 1998;47(2):165–72.
- [5] Levitin G, Lisnianski A. Structure optimization of multi-state system with two failure modes. Reliab Eng Syst Saf 2001;72(1):75–89.
- [6] Liu PX, Zuo MJ, Meng MQH. Using neural network function approximation for optimal design of continuous-state parallel–series systems. Comput Oper Res 2003;30(3):339–52.
- [7] Sakawa M. Multiobjective optimization by the surrogate worth trade-off method. IEEE Trans Reliab 1978;27:311–4.
- [8] Inagaki T, Inoue K, Akashi H. Interactive optimization of system reliability under multiple objectives. IEEE Trans Reliab 1978;27:264–7.
- [9] Park KS. Fuzzy apportionment of system reliability. IEEE Trans Reliab 1987;36:129–32.
- [10] Dhingra AK. Optimal apportionment of reliability and redundancy in series systems under multiple objectives. IEEE Trans Reliab 1992;41(4):576–82.
- [11] Rao SS, Dhingra AK. Reliability and redundancy apportionment using crisp and fuzzy multiobjective optimization approaches. Reliab Eng Syst Saf 1992;37:253–61.
- [12] Huang HZ. Fuzzy multi-objective optimization decision-making of reliability of series system. Microelectron Reliab 1997;37(3):447–9.
- [13] Ravi V, Reddy PJ, Zimmermann HJ. Fuzzy global optimization of complex system reliability. IEEE Trans Fuzzy Syst 2000;8:241–8.
- [14] Huang HZ, Tian Z, Gu Y. Reliability and redundancy apportionment optimization using interactive physical programming. Int J Reliab Qual Saf Eng 2004;11(3):213–22.
- [15] Messac A. Physical programming: effective optimization for computational design. AIAA J 1996;34(1):149–58.
- [16] Messac A, Wilson B. Physical programming for computational control. AIAA J 1998;36(2):219–26.
- [17] Messac A, Martinez M, Simpson T. Introduction of a product family penalty function using physical programming. ASME J Mech Des 2002;124(2):164–72.
- [18] Messac A, Ismail-Yahaya A. Multiobjective robust design using physical programming. Structural and multidisciplinary optimization. J Int Soc Struct Multidiscip Optim (ISSMO) 2002;23(5):357–71.
- [19] Patel M, Lewis KE, Maria A, Messac A. System design through subsystem selection using physical programming. AIAA J 2003;41(6):1089–96.
- [20] Davis L. Handbook of genetic algorithms. New York: Van Nostrand Reinhold; 1991.
- [21] Barlow RE, Wu AS. Coherent systems with multi-state components. Math Oper Res 1978;3(4):275–81.
- [22] Xuan G, Cheng R. Genetic algorithm and engineering design. Beijing: Science Press; 2000.