# Particle Evolutionary Swarm for Design Reliability Optimization

Angel E. Muñoz Zavala, Enrique R. Villa Diharce
and Arturo Hernández Aguirre

Center for Research in Mathematics (CIMAT)
Department of Computer Science
A.P. 402, Guanajuato, Gto. CP 36000, México
aemz,villadi,artha@cimat.mx

**Abstract.** This papers proposes an enhanced Particle Swarm Optimization algorithm with multi-objective optimization concepts to handle constraints, and operators to keep diversity and exploration. Our approach, PESDRO, is found robust at solving redundancy and reliability allocation problems with two objective functions: reliability and cost. The approach uses redundancy of components, diversity of suppliers, and incorporates a new concept called *Distribution Optimization*. The goal is the optimal design for reliability of coherent systems. The new technique is compared against algorithms representative of the state-of-the-art in the area by using a well-known benchmark. The experiments indicate that the proposed approach matches and often outperforms such methods.

## 1 Introduction

The reliability of any device is very important for manufacturers and users. Larger reliability of the final product is desired but, the consequent rise in production cost has negative effects on the user's budget. Therefore, the design reliability optimization problem is phrased as reliability improvement at a minimum cost. The common sense perception of reliability is the absence of failures. Therefore, reliability is sometimes referred to as "quality in time dimension", because is determined by the failures that may or may not occur to the product during its life span. The design reliability problem is hard and challenging, mainly due to the interaction of many subsystems whose conflicting local goals must contribute to the overall performance. New product design involves the specification of performance requirements, the evaluation and selection of components that perform some defined function, and the determination of the system level architecture.

The problems of interest to us are the redundancy and reliability allocation problems. There are two conflicting goals in them: reliability, and cost. The

allocation problem is characterized by a large combinatorial search space, ruled by multiple design constraints.

Several optimization approaches have previously been used to solve the reliability allocation problem [1]. We introduce a new approach based upon the Particle Swarm Optimization (PSO) paradigm which was originally proposed by Kennedy and Eberhart [2]. Our approach, based on multi-objective optimization concepts (treats constraints as objective functions), includes: a selection criteria based on feasibility rules; a local-best PSO with ring topology organization; and two perturbation operators aimed to keep diversity.

The remainder of this paper is organized as follows. In Section 2, we introduce the problem of interest. Section 3 introduces our proposed approach. In Section 4, we describe a benchmark of 3 test functions. Section 5 provided a comparison of results with respect to techniques representative of the state-of-the-art in the area. Finally, our conclusion and future work are provided in Section 6.

## 2 System Reliability Optimization

As noted before, the redundancy and allocation problem deals with the optimization of reliability and cost. Frequently, this problem is described as optimal reliability design, subject to cost and weight constraints. Thus, cost and weight constraints are also optimized during the process. The optimization criterion could be posed in two different forms:

- Maximization of system reliability, subject to cost and weight constraints.
- Minimization of system cost, subject to reliability and weight constraints.

The reliability of a product at time $t$, $R(t)$, is the probability that it works like foresaw, during the time interval $(0, t]$; under several operational conditions and environment. The distribution of the time to failure of a product, determines its $R(t)$ reliability at time $t$.

$$R(t) = 1 - F(t) = P(T > t) \qquad t > 0 \tag{1}$$

where $T$ is the time to failure and $F(t)$ is the cumulative distribution function. $F(t)$ thus denotes the probability that the unit fails within the time interval $(0, t]$.

The reliability of a product with several components is calculated using its structure function. Figure 1 shows the three kind of structures used to model the system's components of this paper.

Each system has its own structure function. The general structure functions for series and parallel structures are shown in Equations 2 and 3, where $n$ is the number of components. The Equation 4 presents the structure function of a k-out-of-n system with $n$ identical components, where $k$ is the minimum number of components required for a system to work.

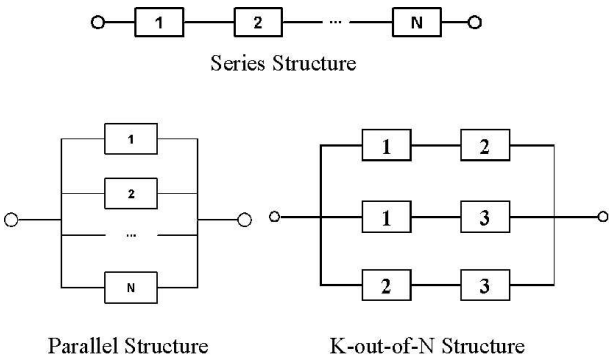


**Fig. 1.** Structures to model system components

– Series

$$R_S = \prod_{i=1}^{n} R_i \tag{2}$$

– Parallel

$$R_S = 1 - \prod_{i=1}^{n} (1 - R_i) \tag{3}$$

– K-out-of-N

$$R_S = \sum_{i=k}^{n} \binom{n}{i} (R)^i (1-R)^{n-i} \tag{4}$$

The diversity of structures, resource constraints, and options for reliability improvement has led to the construction and analysis of several optimization models. We are interested in two general cases: redundancy allocation problem and reliability allocation problem. For the first case, there is a set of discrete components to choose from, whose characteristics are known (reliability or distribution function, cost, weight, etc.). The objectives of this combinatorial problem are the selection of components to use, and the corresponding redundancy levels. The redundancy allocation problem has been shown NP-hard by Chern [3]. For the second case, the component reliability or a vector of distribution parameters is treated as the design variables, and component's cost is a predefined increasing function of the component reliability.

As noted, our design optimization problem may appear in two forms:

**P1**

$$\text{Find} \quad \boldsymbol{R} \text{ which maximize } R_s \tag{5}$$

subject to:

$$\sum_{i=1}^{n} C_i \leq c \tag{6}$$

$$\sum_{i=1}^{n} W_i \leq w \tag{7}$$

**P2**

$$\text{Find } \boldsymbol{R} \text{ which minimize } C_s \tag{8}$$

subject to:

$$R_s \geq r \tag{9}$$

$$\sum_{i=1}^{n} W_i \leq w \tag{10}$$

where $\boldsymbol{R}$ is the reliability vector of each component, $\boldsymbol{R} = (R_1, R_2, \ldots, R_n)$, $C_i$ and $W_i$ are the cost and the weight of the ith component, $n$ is the number of components, and $c, w, r$ are constants: $c \geq 0$, $w \geq 0$, and $0 \leq r \leq 1$.

There are several assumptions we must consider to solve the redundancy - reliability allocation problem:

- The state of the components and the system have only two options: good or bad.
- Failures of components are independent events.
- Failed components do not damage the system.
- Failed components are not repaired.
- The component's reliability or distribution function is known.

All redundancy and reliability allocation problems meeting these assumptions are denominated *coherent systems*. A system of components is said to be coherent if all its components are relevant and the state of the system is nondecreasing in each argument. One might get the impression that all systems of interest must be coherent, but this is not the case.

A component $i$ is irrelevant if for all combinations of the state components, the state of the $i$th component does not affect the state of the system.

Here we will assume that a system will not run worse than before when we replace a component in failed state with one that is operating correctly. This is the same as requiring that the state of the system is nondecreasing in each argument.

## 3 The PESDRO Algorithm

A solution algorithm for this problem is based on PSO search. The PSO algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. In PSO, individuals, referred to as particles, are "flown" through a hyperdimensional search space. PSO is a kind of

symbiotic cooperative algorithm, because the changes to the position of particles within the search space are based on the social-psychological tendency of individuals to emulate the success of other individuals.

The feature that drives PSO is social interaction. Individuals (particles) within the swarm learn from each other, and based on this shared knowledge tend to become more similar to their "better" neighbors. A social structure in PSO is determined through the formation of neighborhoods. These neighborhoods are determined through labels attached to every particle in the flock (so it is not a topological concept). Thus, the social interaction is modeled by spreading the influence of a "global best" all over the flock as well as neighborhoods are influenced by the best neighbor and their own past experience.

Figure 2 shows some neighborhood structures that have been proposed and studied [4]. Our approach, PESDRO, adopts the **ring** topology. In the ring organization, each particle communicates with its $n$ immediate neighbors. For instance, when $n = 2$, a particle communicates with its immediately adjacent neighbors as illustrated in Figure 2(b). The neighborhood is determined through an index label assigned to all individuals. This version of the PSO algorithm is referred to as *lbest* (LocalBest). It should be clear that the ring neighborhood structure properly represents the LocalBest organization. It has the advantage that a larger area of the search space is traversed, favoring search space exploration (although convergence has been reported slower) [5,6].

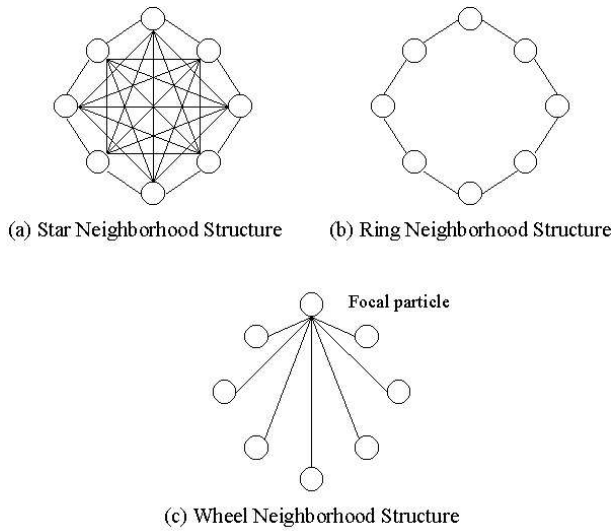


**Fig. 2.** Neighborhood structures for PSO [4]

PSO-LocalBest has been reported to excel over other topologies when the maximum velocity is restricted. PESDRO's results with and without restricted

velocity reached similar conclusions noted by Franken and Engelbretch [7], thus, PESDRO algorithm incorporates this feature. Figure 3 shows the standard PSO algorithm adopted for our approach, where $X_L$ and $X_U$ are the limits of the search space, and $n$ is the population size. The pseudo-code of **LocalBest** function is shown in Figure 4.

```
P0 = Rand(XL, XU)
F0 = Fitness ( P0 )
PBest0= P0
FBest0= F0
Do While i¡MaxGenerations
  LBesti = LocalBest ( PBesti, FBesti )
  Si = Speed ( Si, Pi, PBesti, LBesti)
  Pi+1 = Pi + Si
  Fi+1 = Fitness ( Pi+1 )
  For k = 0 To n
    < PBesti+1[k], FBesti+1[k] > = Best ( FBesti[k], Fi+1[k] )
  End For
End Do
```

**Fig. 3.** Pseudo-code of $PSO$ algorithm with local best

Constraint handling is embedded into the selection operator, and described by "feasibility and dominance" rules. These rules are: 1) given two feasible particles, pick the non-dominated one; 2) if both particles are infeasible, pick the particle with the lowest sum of constraint violation, and 3), given a pair of feasible and infeasible particles, the feasible particle wins.

```
For k = 0 To n
  LBesti[k] = Best(FBesti[k − 1], FBesti[k + 1])
End For
```

**Fig. 4.** Pseudo-code of **LocalBest**($PBest_i$,$FBest_i$)

The *speed* vector drives the optimization process and reflects the socially exchanged information. Figure 5 shows the pseudo-code of **Speed** function, where $c1 = 0.1$, $c2 = 1$, and $w$ is the inertia weight. The inertia weight controls the influence of previous velocities on the new velocity.

```
For k = 0 To n
  For j = 0 To d
    r1 = c1 * U(0, 1)
    r2 = c2 * U(0, 1)
    w = U(0.5, 1)
    S_i[k, j] = w * S_i[k, j] +
      r1 * (PBest_i[k, j] - P_i[k, j]) +
      r2 * (LBest_i[k, j] - Pbest_i[k, j])
  End For
End For
```

**Fig. 5.** Pseudo-code of **Speed**($S_i$, $P_i$, $PBest_i$, $LBest_i$)

### 3.1 Perturbation operators

PESDRO algorithm makes use of two perturbation operators to keep diversity and exploration. PESDRO has three stages; in first stage the standard PSO algorithm [6]is performed, then the perturbations are applied in the next two stages.

The main algorithm of PESDRO is shown in Figure 6.

The goal of the second stage is to add a perturbation in a way similar to the so called "reproduction operator" found in *differential evolution* algorithm. This perturbation, called C-Perturbation, is applied all over the flock to yield a set of temporal particles $Temp$. Each member of the $Temp$ set is compared with the corresponding (father) member of $PBest_{i+1}$, so the perturbed version replaces the father if it has a better fitness value. Figure 7 shows the pseudo-code of the **C-Perturbation** operator.

In the third stage every vector is perturbed again so a particle could be deviated from its current direction as responding to external, maybe more promissory, stimuli. This perturbation is performed with some probability on each dimension of the particle vector, and can be explained as the addition of random values to each particle component. The perturbation, called M-Perturbation is applied to every particle in the current population to yield a set of temporal particles $Temp$. Again, as for C-Perturbation, each member of $Temp$ is compared with its corresponding (father) member of the current population, and the better one wins. Figure 8 shows the pseudo-code of the **M-Perturbation** operator. The perturbation is performed with probability $p = 1/d$, where $d$ is the dimension of the decision variable vector.

These perturbations, in differential evolution style, have the advantage of keeping the self-organization potential of the flock as no separate probability distribution needs to be computed [8]. Zhang and Xie also try to keep the self-organization potential of the flock by applying mutations (but only) to the particle best (in their DEPSO system) [9]. In PESDRO, the self-organization is not broken as the link between father and perturbed version is not lost. Thus, the

$P_0$ = Rand($X_L$, $X_U$)
$F_0$ = Fitness ( $P_0$ )
$PBest_0$= $P_0$
**Do While i¡MaxGenerations**
  $LBest_i$ = **LocalBest** ( $PBest_i$, $FBest_i$ )
  $S_i$ = **Speed** ( $S_i$, $P_i$, $PBest_i$, $LBest_i$)
  $P_{i+1}$ = $P_i$ + $S_i$
  $F_{i+1}$ = Fitness ( $P_{i+1}$ )
  **For** $k$ = 0 **To** $n$
    < $PBest_{i+1}[k], FBest_{i+1}[k]$ > = Best ( $FBest_i[k]$, $F_{i+1}[k]$ )
  **End For**
  $Temp$ = **C − Perturbation** ($P_{i+1}$ )
  $FTemp$ = Fitness ( $Temp$ )
  **For** $k$ = 0 **To** $n$
    < $PBest_{i+1}[k], FBest_{i+1}[k]$ > = Best ( $PBest_{i+1}[k]$ , $FTemp$[k] )
  **End For**
  $Temp$ = **M − Perturbation** ($P_{i+1}$ )
  $FTemp$ = Fitness ( $Temp$ )
  **For** $k$ = 0 **To** $n$
    < $PBest_{i+1}[k], FBest_{i+1}[k]$ > = Best ( $PBest_{i+1}[k]$ , $FTemp$[k] )
  **End For**
  $P_i$ = $P_{i+1}$
**End Do**

**Fig. 6.** Main algorithm of $PESDRO$

perturbation can be applied to the entire flock. Note that these perturbations are suitable for real-valued function optimization.

### 3.2 Important aspects of PESDRO

Particle Evolutionary Swarm for Design Reliability Optimization (PESDRO) is an implementation of PSO to solve the redundancy and reliability allocation problem. We propose a new method that offers a variety of options to optimize each component of a system. Several approaches representatives of the state-of-the-art to solve the redundancy and reliability allocation problem, seek the optimum by applying only one approach, either redundancy or reliability, to all system's components. In this paper, we propose a combined approach for the optimization of each component, either by redundancy or by reliability allocation, but yielding a system with optimum reliability and costs.

Redundancy and diversity techniques are used in redundancy allocation problem.

- Redundancy: It is a technique that replaces one component by a subsystem formed by N equal components with a parallel structure. Subsystem's reliability is increased by each component allocated in parallel. Is important

```
For k = 0 To n
  For j = 0 To d
    r = U(0, 1)
    p1 = Random(n)
    p2 = Random(n)
    p3 = Random(n)
    Temp[k, j] = P_{i+1}[p1, j] + r * (P_{i+1}[p2, j] - P_{i+1}[p3, j])
  End For
End For
```

**Fig. 7.** Pseudo-code of $\mathbf{C-Perturbation}(P_{i+1})$

```
For k = 0 To n
  For j = 0 To d
    r = U(0, 1)
    If r ≤ 1/d Then
      Temp[k, j] = Rand(LI, LS)
    Else
      Temp[k, j] = P_{i+1}[k, j]
  End For
End For
```

**Fig. 8.** Pseudo-code of $\mathbf{M-Perturbation}(P_{i+1})$

to mention that there is only one supplier for the component, therefore, a subsystem is conformed by $n$ components with same reliability, cost and weight.

- Diversity: One component has several suppliers but it is not limited to choose only one. Instead, it is free to build a subsystem of N components with parallel structure, whose components come from different suppliers.

Reliability allocation problem could be solved by using the technique proposed in this paper, called distribution optimization.

- Distribution Optimization: The goal is to increase the component's reliability until a expected or required value is met. Component's cost is a predefined increasing function of the component reliability.

$$g(R_i) = a\,[\Delta R_i(t)]^b \tag{11}$$

$$\Delta R_i(t) = RF_i(t) - RI_i(t) \tag{12}$$

When the reliability function is independent of the time, a constant reliability value is used as the initial reliability $RI_i(t)$; and the final reliability $RF_i(t)$ is used to improve the reliability component until its desired value.

But, if the reliability function of the component is dependent of the time, then when a final reliability value $RF_i(t)$ is reached, we must find the optimal distribution parameters such that the distribution yields that value. We try to find the set of distribution parameters that minimize the distance with respect to the original distribution parameters, and yields a final reliability value $RF_i(t)$.

For instance, in Figure 9, an initial reliability value is at Weibull($\alpha = 70, \beta = 2$) that yields a reliability equal to $RI(50) = 0.60$, but the new parameters $\alpha, \beta$ must be determined for the yielded reliability curve $RF(50) = 0.77$. There are two main cases that may occur. In the first case, the distribution function has only one parameter, then the relation is one-one between the reliability function and its parameter; also there is one and only one parameter value that yields the desire reliability at time $t = 50$.

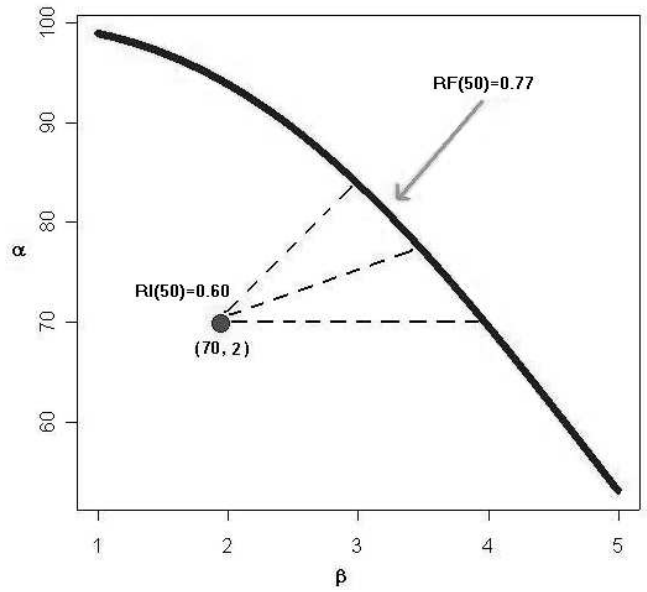


**Fig. 9.** Finding optimal distribution parameters.

In the second case, the number of distribution parameters is at least two, $p > 2$, then there are an infinity distribution parameters set that yields the same reliability value. For this case, we can find the new distribution parameters set that has the minimum distance with respect to the original distribution parameters, and even standardize the distance of each parameter. Also, we can change only one parameter and set the other $p - 1$ distribution parameters, then the first case appears.

Starting off from the mentioned techniques, a design optimization tool could be created to simultaneously solve both redundancy and reliability allocation problems. In PESDRO, we assign a sub-population to each component.

## 4 Experiments

PESDRO was evaluated using three well-known problems, the first two problems are of type **P1**, and the third one is type **P2**. The results are contrasted against a Genetic Algorithm, an Ant Colony System, and Tabu Search. The problems are explained next.

1. **Test case 1**: The first test problem was originally proposed by Fyffe, Hines and Lee in 1968 [10] and modified by Nakagawa and Miyazaki in 1981 [11]. Fyffe, Hines and Lee specified a system with 14 subsystems. For each subsystem, there are three or four component options. Component cost, weight and reliability are provided in Table 1. The objective is to maximize system reliability given constraints limits of 130 units of system cost, 170 units of system weight. The maximum number of components within a subsystem has been defined to be six ($n_{max,i} = 6$). Results are show in Table 4 and the analysis is presented in Section 5.1.

| | $S_1$ | | | $S_2$ | | | $S_3$ | | | $S_4$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i | $R_{i1}$ | $C_{i1}$ | $W_{i1}$ | $R_{i2}$ | $C_{i2}$ | $W_{i2}$ | $R_{i3}$ | $C_{i3}$ | $W_{i3}$ | $R_{i4}$ | $C_{i4}$ | $W_{i4}$ |
| 1 | 0.90 | \$1 | 3 | 0.93 | \$1 | 4 | 0.91 | \$2 | 2 | 0.95 | \$2 | 5 |
| 2 | 0.95 | \$2 | 8 | 0.94 | \$1 | 10 | 0.93 | \$1 | 9 | | | |
| 3 | 0.85 | \$2 | 7 | 0.90 | \$3 | 5 | 0.87 | \$1 | 6 | 0.92 | \$4 | 4 |
| 4 | 0.83 | \$3 | 5 | 0.87 | \$4 | 6 | 0.85 | \$5 | 4 | | | |
| 5 | 0.94 | \$2 | 4 | 0.93 | \$2 | 3 | 0.95 | \$3 | 5 | | | |
| 6 | 0.99 | \$3 | 5 | 0.98 | \$3 | 4 | 0.97 | \$2 | 5 | 0.96 | \$2 | 4 |
| 7 | 0.91 | \$4 | 7 | 0.92 | \$4 | 8 | 0.94 | \$5 | 9 | | | |
| 8 | 0.81 | \$3 | 4 | 0.90 | \$5 | 7 | 0.91 | \$6 | 6 | | | |
| 9 | 0.97 | \$2 | 8 | 0.99 | \$3 | 9 | 0.96 | \$4 | 7 | 0.91 | \$3 | 8 |
| 10 | 0.83 | \$4 | 6 | 0.85 | \$4 | 5 | 0.90 | \$5 | 6 | | | |
| 11 | 0.94 | \$3 | 5 | 0.95 | \$4 | 6 | 0.96 | \$5 | 6 | | | |
| 12 | 0.79 | \$2 | 4 | 0.82 | \$3 | 5 | 0.85 | \$4 | 6 | 0.90 | \$5 | 7 |
| 13 | 0.98 | \$2 | 5 | 0.99 | \$3 | 5 | 0.97 | \$2 | 6 | | | |
| 14 | 0.90 | \$4 | 6 | 0.92 | \$4 | 7 | 0.95 | \$5 | 6 | 0.99 | \$6 | 9 |

**Table 1.** Component's data for Test case 1

2. **Test case 2**: Coit and Liu in 2000 [12] proposed a system made of 14 *k-out-of-n* subsystems with $k_i \in \{1, 2, 3\}$. The problem is a modified version of the original problem proposed by Fyffe, Hines and Lee [10] and modified

by Nakagawa and Miyazaki [11]. For each subsystem, there are three or four component options. Component cost, weight and exponential distribution parameter ($\lambda_{ij}$) are given in Table 2. The objective is to maximize system reliability at a time of 100 hours given constraints limits of 130 units of system cost, 170 units of system weight. The maximum number of components within a subsystem has been defined to be six ($n_{max,i} = 6$). Results are show in Table 5 and the analysis is presented in Section 5.2.

| | | $S_1$ | | | $S_2$ | | | $S_3$ | | | $S_4$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i | $k_i$ | $\lambda_{i1}$ | $C_{i1}$ | $W_{i1}$ | $\lambda_{i2}$ | $C_{i2}$ | $W_{i2}$ | $\lambda_{i3}$ | $C_{i3}$ | $W_{i3}$ | $\lambda_{i4}$ | $C_{i4}$ | $W_{i4}$ |
| 1 | 1 | 0.001054 | $1 | 3 | 0.000726 | $1 | 4 | 0.000943 | $2 | 2 | 0.000513 | $2 | 5 |
| 2 | 2 | 0.000513 | $2 | 8 | 0.000619 | $1 | 10 | 0.000726 | $1 | 9 | | | |
| 3 | 1 | 0.001625 | $2 | 7 | 0.001054 | $3 | 5 | 0.001393 | $1 | 6 | 0.000834 | $4 | 4 |
| 4 | 2 | 0.001863 | $3 | 5 | 0.001393 | $4 | 6 | 0.001625 | $5 | 4 | | | |
| 5 | 1 | 0.000619 | $2 | 4 | 0.000726 | $2 | 3 | 0.000513 | $3 | 5 | | | |
| 6 | 2 | 0.000101 | $3 | 5 | 0.000202 | $3 | 4 | 0.000305 | $2 | 5 | 0.000408 | $2 | 4 |
| 7 | 1 | 0.000943 | $4 | 7 | 0.000834 | $4 | 8 | 0.000619 | $5 | 9 | | | |
| 8 | 2 | 0.002107 | $3 | 4 | 0.001054 | $5 | 7 | 0.000943 | $6 | 6 | | | |
| 9 | 3 | 0.000305 | $2 | 8 | 0.000101 | $3 | 9 | 0.000408 | $4 | 7 | 0.000943 | $3 | 8 |
| 10 | 3 | 0.001863 | $4 | 6 | 0.001625 | $4 | 5 | 0.001054 | $5 | 6 | | | |
| 11 | 3 | 0.000619 | $3 | 5 | 0.000513 | $4 | 6 | 0.000408 | $5 | 6 | | | |
| 12 | 1 | 0.002357 | $2 | 4 | 0.001985 | $3 | 5 | 0.001625 | $4 | 6 | 0.001054 | $5 | 7 |
| 13 | 2 | 0.000202 | $2 | 5 | 0.000101 | $3 | 5 | 0.000305 | $2 | 6 | | | |
| 14 | 3 | 0.001054 | $4 | 6 | 0.000834 | $4 | 7 | 0.000513 | $5 | 6 | 0.000101 | $6 | 9 |

**Table 2.** Component's data for Test case 2

3. **Test case 3**: Coit and Smith in 1996 [13] proposed a system made of 2 *k-out-of-n* subsystems with $k_1 = 4$ and $k_2 = 2$. For each subsystem, there are ten component options. Component cost, weight and reliability are given in Table 3. The objective is to minimize system cost given constraints limits of 650 units of system weight and 0.975 units of system reliability. The maximum number of components within a subsystem has been defined to be eight ($n_{max,i} = 8$). Results are show in Table 6 and the analysis is presented in section 5.3.

## 5 Comparison of Results

Because of the stochastic nature of PESDRO, 30 trials with 100 particles were performed. The best solution of each run is stored and later used to compute the statistics. PESDRO was compared to other algorithms used to solve the experiments explained in Section 4 (information regarding experiment conditions of other authors are not available as to perform an exact comparison).

### 5.1 Test case 1

For this test, 30 runs with 3333 generations each were performed. Table 4 presents the corresponding results from the Ant Colony System of Liang and

| i | 1 ($k_1 = 4$) | | | 2 ($k_2 = 2$) | | |
|---|---|---|---|---|---|---|
| | $R_{1j}$ | $C_{1j}$ | $W_{1j}$ | $R_{2j}$ | $C_{2j}$ | $W_{2j}$ |
| S1 | 0.981 | \$95 | 52 | 0.931 | \$137 | 83 |
| S2 | 0.933 | \$86 | 94 | 0.917 | \$132 | 96 |
| S3 | 0.730 | \$80 | 32 | 0.885 | \$127 | 94 |
| S4 | 0.720 | \$75 | 92 | 0.857 | \$122 | 93 |
| S5 | 0.708 | \$61 | 41 | 0.836 | \$100 | 95 |
| S6 | 0.699 | \$45 | 33 | 0.811 | \$59 | 63 |
| S7 | 0.655 | \$40 | 98 | 0.612 | \$54 | 65 |
| S8 | 0.622 | \$36 | 96 | 0.432 | \$41 | 49 |
| S9 | 0.604 | \$31 | 83 | 0.389 | \$36 | 33 |
| S10 | 0.352 | \$26 | 66 | 0.339 | \$30 | 51 |

**Table 3.** Component's data for Test case 3

Smith (1999) [14], where 10 runs of the algorithm were performed. Also shown are the results of the TSRAP algorithm of Kulture-Konak, Smith and Coit (2003) [15] (10 trials of the algorithm were performed).

| Reliability | AS 10 runs | TSRAP 10 runs | PESDRO 30 runs |
|---|---|---|---|
| Maximum | 0.963510 | 0.970760 | 0.966950 |
| Minimum | 0.959090 | - | 0.952045 |
| Average | 0.962160 | - | 0.960001 |
| Std. Dev. | 0.001790 | 0.000490 | 0.003677 |

**Table 4.** Test case 1: A comparison of AS [14], TSRAP [15] and PESDRO

### 5.2 Test case 2

For this test problem, 30 runs with 3333 generations each were performed . Table 5 presents the corresponding results from TSRAP of Kulture-Konak, Smith and Coit (2003) [15] (10 trials of the algorithm were performed).

| Reliability | TSRAP 10 runs | PESDRO 30 runs |
|---|---|---|
| Maximum | 0.413450 | 0.403942 |
| Minimum | - | 0.341003 |
| Average | - | 0.370370 |
| Std. Dev. | 0.002625 | 0.015551 |

**Table 5.** Test case 2: A comparison of TSRAP [15] and PESDRO

### 5.3 Test case 3

For this last problem, 30 runs with 1666 generations each were performed. Table 6 presents the corresponding results from TSRAP algorithm of Kulture-Konak, Smith and Coit (2003) [15], where 20 trials of the algorithm were performed. Also shown are the results of the Genetic Algorithm of Coit and Smith (1996) [13](20 trials of the algorithm were performed).

| Cost | GA 20 runs | TSRAP 20 runs | PESDRO 30 runs |
|---|---|---|---|
| Minimum | \$727.00 | \$727.00 | \$727.00 |
| Maximum | - | - | \$728.00 |
| Average | \$727.25 | \$727.80 | \$727.27 |
| Std. Dev. | - | - | 0.449776 |

**Table 6.** Test 3: A comparison of the GA [13], TSRAP [15] and PESDRO

The results of PESDRO in the first two test problems are close to the TSRAP's results. For the last test problem, the results of PESDRO are competitive with the results of the TSRAP and the GA. PESDRO algorithm has a good performance because it works in two steps. First, PESDRO quickly finds feasible solutions by means of the perturbation operators (coarse search). At the same time, those operators help the algorithm to avoid stagnation at local optima. When a feasible solution is near the optimal, the search is mainly driven by the standard PSO operators, thus performing a fine search.

## 6 Conclusions and Future Work

In this paper a PSO approach was described and applied to the optimal design of redundancy and reliability allocation problems. PESDRO was demonstrated in three test problems with competitive results. A new technique is propose to solve reliability allocation problem. There are many applications for *Distribution Optimization* in system reliability optimization [16]. PESDRO provided results competitive with TSRAP in all problems, it is better than Ant System in Test problem 1, and in the average is better than TSRAP in problem 3. The proposed system can deal with 4 different probability distributions, and any combination of parallel, series, and K-out-of-N subsystems. This ability makes the system applicable to a broad kind of problems; this feature is not found in the other reviewed approaches: TSRAP, AS, or GA. Thus, the trade-off between generalization and specialization must be considered before any analysis of the results.

## References


1. Kuo,W., Prasad,R.: An Annotated Overview of System Reliability Optimization. *IEEE Transactions on Reliability*, Vol. **49**(2) (June 2000) 176-187.
2. Kennedy,J., Eberhart,R.: Particle Swarm Optimization. *Proceedings of the IEEE International Conference On Neural Networks*, Vol. **4** (1995) 1942-1948.
3. Chern,M.: On the Computational Complexity of Reliability Redundancy Allocation in a Series System. *Operations Research Letters*, Vol. **11** (1992) 309-315.
4. Kennedy,J.: Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance. *IEEE Congress on Evolutionary Computation*, Vol. **3** (1999) 1931-1938.
5. Eberhart,R., Dobbins,R., Simpson,P.: Computational Intelligence PC Tools. Academic Press, (1996).
6. Kennedy,J., Eberhart,R.: The Particle Swarm: Social Adaptation in Information-Processing Systems. *New Ideas in Optimization*, McGraw-Hill (1999) 379-387.
7. Franken, N. and Andries P. Engelbrecht. Comparing PSO structures to learn the game of checkers from zero knowledge. In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, Canberra, Australia, (2003) Vol. 1, pages 234-241
8. Storn, R. Sytem Design by Constraint Adaptation and Differential Evolution. *IEEE Trans. on Evolutionary Computation*, 1999, Vol. 3, No. 1, pp. 22 - 34
9. Zhang, W J., Xie XF. DEPSO: Hybrid Particle Swarm with Differential Evolution Operator. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, Washington D.C., USA, (2003) 3816-3821
10. Fyffe,D., Hines,W., Lee,N.: System reliability allocation and a computational algorithm. *IEEE Transactions on Reliability*, Vol. **17** (1968) 74-79.
11. Nakagawa,Y., Miyazaki,S.: Surrogate Constraints Algorithm for Reliability Optimization Problems with Two Constraints. *IEEE Transactions on Reliability*, Vol. **30** (1981) 175-180.
12. Coit,D., Liu,J.: System Reliability Optimization with k-out-of-n Subsystems. *International Journal of Reliability, Quality and Safety Engineering*, Vol. **7**(2) (2000) 129-143.
13. Coit,D., Smith,A.: Reliability Optimization of Series - Parallel Systems Using a Genetic Algorithm. *IEEE Transactions on Reliability*, Vol. **45**(2) (1996) 254-260.
14. Liang,Y., Smith,A.: An Ant System Approach to Redundancy Allocation. *Proceeding of the 1999 Congress on Evolutionary Computation, IEEE, Piscataway,N.y.*, (1999) 1478-1482.
15. Kulturel-Konak,S., Smith,A., Coit,D.: Efficiently Solving the Redundancy Allocation Problem Using Tabu Search. *IIE Transactions*, Vol. **35** (2003) 515-526.
16. Angel E. Muñoz Zavala. Optimal Design for Reliability. Master Thesis in Computer Science and Industrial Mathematics, Center for Research in Mathematics, 2004