Adaptive Genetic Local Search Algorithms for Solving Reliability Optimization Problems

Minoru Mukuda* Member
YoungSu Yun** Non-member
Mitsuo Gen*** Member

This paper proposes an adaptive genetic local search (aGLS) algorithm for effectively solving reliability optimization problems. The proposed aGLS hybridizes a local search technique and genetic algorithm (GA). The local search technique is incorporated into the GA loop and its scheme is adaptively regulated according to the change of the average fitness value at every generation in GA.

For more various comparisons with the proposed aGLS, conventional GLS algorithm with local search that does not use any adaptive scheme is also suggested. These two algorithms are tested and analyzed using two complex reliability optimization problems. Numerical result shows that the proposed aGLS outperforms the conventional HGA.

Keywords: genetic local search, reliability optimization problem, genetic algorithm, local search

1. Introduction

In the past few decades, reliability optimization problems including allocation of redundancy problems, reliability apportionment problems, and redundancy allocation problems have been treated by many researchers. Furthermore, the practicality in recent process synthesis and process optimization of highly reliable systems leads to the extended system reliability design considering to locate the optimum levels of component reliability and the number of redundant components at each subsystem. This problem, called as an optimal reliability assignment/redundant allocation problem, is more difficult reliability apportionment or redundancy allocation problems because it should be simultaneously considered complex search space. Recently genetic algorithms (GAs) have been proved to be a more effective approach for these reliability optimization problems.

However, conventional GA approaches have some weaknesses in its application. That is, pure GA-based approaches may converge to a local optimal solution prematurely before locating the global optimal solution because of their fundamental requirement. These approaches have also a weakness in taking too much time to adjust fine-tuning structure of the GA parameters (i.e., crossover rate, mutation rate, and others). Therefore, these two kinds of “blindness” may prevent them from being really of practical interest to many complex optimization problems such as reliability optimization problems mentioned above. To improve these weaknesses of GA-based approach, various hybrid methodologies using conventional heuristics and adaptive schemes have been developed. By applying the conventional heuristics and the adaptive schemes to GA, recently hybridized GAs are more effective and more robustness than pure GA-based approach or other conventional heuristics.

Based on these contributions to GA, this paper proposes a new hybrid algorithm, adaptive genetic local search (a-GLS) algorithm, with a local search technique and an adaptive scheme. For various comparisons with the proposed a-GLS, the conventional GLSs with the local search technique that does not use any adaptive scheme is also suggested. These two GLSs are tested and analyzed in numerical examples using reliability optimization problems.

2. Reliability Optimization Problems

Reliability optimization problems are usually decomposed into functional entities composed of units, subsystems, or components for the purpose of reliability. Combinatorial aspects of reliability analysis are connected to the components neither purely in series nor purely in parallel.

The objective is to determine the optimum level of component reliability and the number of redundant components at each subsystem while meeting the goal with a maximum reliability. Here, we represent the optimal reliability assignment / redundant allocation problem into the general form of nonlinear mixed integer programming (nMIP). We have to find the number of components in each subsystem as integer decision variables $x = [x_1, x_2, \ldots, x_n]$ and to determined levels of component reliability in each subsystem as continuous decision variables $r = [r_1, \ldots, r_n]$

$$\max f(x, r) = \sum_{i=1}^{n} f_i(x_i, r_i)$$

s. t. $g_i(x, r) = \sum_{j=1}^{q} g_{ij}(x_j, r_j) \leq b_i, \quad i = 1, 2, \ldots, q$

$r_j \leq r_j \leq r_j'$ : real, $j = 1, 2, \ldots, n$

$x_j \leq x_j \leq x_j'$ : integer, $j = 1, 2, \ldots, n$
where $f_j(x_i, r_j)$ is the $j$-th non-linear objective function represented a system reliability, $g_i(x_i, r_j)$ is the $j$-th non-linear function on the $i$-th constraint represented a system resource restraint, $b_i$ is the $i$-th right-hand side constant or available resource, $x_i^l$ and $x_i^u$ are the lower and upper bounds for the integer decisions variable $x_i$, $r_j^l$ and $r_j^u$ are the lower and upper bounds for the continuous decisions variable $r_j$, respectively. $j$ is numbers of subsystem.

3. Design of Adaptive Genetic Local Search Algorithm

In this section, we first mention GA approach and local search one as a general type, and then suggest the concept of the conventional GLS algorithm and proposed aGLS one.

3.1 Genetic Algorithm Approach The main role of GA approach, when constructing GLS, is to perform global search within all feasible search spaces. For this purpose, GAs have particular mechanisms for the global search: i) population-oriented search scheme for locating more various individuals, ii) three genetic operators (selection, crossover and mutation) for performing more various changes within population.

3.2 Local Search Approach GA can do global search in entire space but there are no ways for local search around the convergence area generated by GA loop, thus GA is sometimes impossible or insufficient finding optimum in the problems requiring complex and precision values. To overcome the weakness, various methods for hybridizing GA using conventional local search techniques have been suggested.

One of the common forms of hybridized GA is to incorporate a local search technique to a conventional GA loop. With this hybrid approach, local search technique is applied to each newly generated offspring to move it to a local optimum before injecting it into the new population.

In our approach, we incorporate the iterative hill climbing technique to GA loop. Thus GA carries out global search and the iterative hill climbing carries out local search around the convergence area by GA loop. This technique can guarantee the desired properties of a local search technique for hybridization. The detailed procedure of the technique for minimization problem, is given as follows:

**Procedure: Iterative hill climbing technique in the GA loop**

begin
  select the optimum string $v_o$ in current GA loop;
  generate randomly as many strings as the population size in the neighborhood of the $v_o$, and then calculate the fitness values of the generated new strings by the objective function $f$;
  select the string $v_o$ with the optimal value among the fitness values of the new strings;
  if $f(v_o) > f(v_e)$ then
      $v_e \leftarrow v_o$
  end
end

3.3 GLS with Non-adaptive Local Search (na-GLS)

The na-GLS is a standard type of hybrid algorithm and combines a GA with a local search technique. In this na-GLS, the local search technique is performed as many generation number as the GA loop. Thus, the total iteration number in the na-GLS loop becomes two times larger than that of the GA loop. The total running time of the na-GLS may be also increased.

3.4 GLS with Adaptive Local Search (a-GLS) The a-GLS works with the same operators as the na-GLS. Major difference between the na-GLS and a-GLS is that the a-GLS is adapted in response to recent performance as the approach converges to the optimal solution. For this scheme, we use the average fitness values from continuous two generations of GA loop, and then calculate the fitness value ratio (FVR) at the generations as follow:

$$\text{FVR}(t) = \frac{f_{\text{new}}_{\text{unr}(t)}}{f_{\text{new}}_{\text{unr}(t-1)}}$$

where $f_{\text{new}}_{\text{unr}(t)}$ : average fitness value of new population resulting from elitist selection strategy using parent and offspring sizes at generation $t$.

By this FVR$(t)$, the local search technique will be incorporated into GA loop adaptively. The adapting strategy for minimization problem is as follows:

$$\begin{cases}
\text{do GA with iterative hill climbing technique,} & \text{if FVR}(t) > 1 \\
\text{do GA,} & \text{otherwise}
\end{cases}$$

The adapting strategy of equation (2) means that the local search using the iterative hill climbing technique should be applied in GA loop to increase the locating chance of more respective offspring, if the average fitness value of current generation is not superior to that of the previous one. By this conditional scheme, a-GLS can adaptively determine the use of local search during GA is converged to the optimal solution. Applying the adaptively local search to GA loop is a main scheme of the a-GLS. Therefore, the total iteration number and running time of the a-GLS will be more decreased than those of the na-GLS.

4. GLSs for Experimental Comparison

In this Section, we propose the detailed implementation procedure of the na-GLS and a-GLS. First, for more various comparisons, the canonical GA procedure is presented, and then the procedures of two GLSs are proposed.

4.1 Canonical GA (CGA) For representing the population of the CGA, we use real-number representation instead of bit-string ones. The real-number representation has several advantages of (i) being better adapted to numerical optimization such as reliability optimization problems, (ii) speeding up the search over the bit-string representation, and (iii) easing the development of approaches for hybridizing with other conventional heuristics.

The detailed heuristic procedure for implementing the CGA loop is as follows:

**Step 1: Initial population**

The initial population of the CGA is obtained by random generation.

**Step 2: Genetic operators**

Selection: elitist strategy in enlarged sampling space
Crossover: uniform arithmetic crossover operator
Mutation: uniform mutation operator

**Step 3: Stop condition**

If a pre-defined maximum generation number is reached or an optimal solution is located during genetic search process, then stop; otherwise, go to Step 2.
4.2 na-GLS  The procedure of the na-GLS is to combine the CGA with the iterative hill climbing technique. Its combined heuristic procedure is as follows:

**Steps 1-2:** apply the same steps with the Steps 1 and 2 in the CGA loop.

**Step 3:** apply the iterative hill climbing technique suggested in Section 3-2.

**Step 4:** apply the same step with the Step 3 in the CGA loop.

Fig. 1 shows the flowchart of the na-GLS for minimization problem.

4.3 a-GLS  For the procedure of the a-GLS, Steps 1, 2 and 4 of Section 4.2 are used here. However, in Step 3, the iterative hill climbing technique controlled by the equation (2) of Section 3.4 is used. Fig. 2 shows the flowchart of the procedure of the a-GLS, when minimization is assumed.

5. Numerical Examples

In this Section, two complex reliability optimization problems with the redundancy allocation types of components are suggested. For GA implementation under a same condition, we set the GA parameters as follow:

- Maximum generation number: 5,000
- Population size: 20
- Crossover rate: 0.5
- Mutation rate: 0.1
- Search range for the iterative hill climbing technique: 0.6
- Allogether 30 iterations with different initial random seeds for GA population are executed.

5.1 Test Problem 1 (T-1)  The presented example is an optimal redundancy allocation problem taken from Rabi, Murty, and Reddy (1997). The schematic is in Fig. 3.

The mathematical formulation is as follows:

\[
\begin{align*}
\max & \quad f(x) = \prod_{j=1}^{15} \{1-(1-r_j)^{x_j}\} \\
\text{s.t.} & \quad g_1(x) = \sum_{j=1}^{15} (c_j \cdot x_j) \leq 400 \\
& \quad g_2(x) = \sum_{j=1}^{15} (w_j \cdot x_j) \leq 414 \\
& \quad 1 \leq x_j \leq 10 \quad \text{integer, } j=1,\ldots,15
\end{align*}
\]

where, \(f(x)\) is system reliability, \(g_1(x)\) is total cost for the system, \(x_j\) is the decision variable which is the number of units in the \(j\)-th subsystem, \(g_2(x)\) is total physical limitation for the system, \(w_j\) is physical limitation (size or weight, etc.) of the \(j\)-th subsystem, \(r_j\) and \(c_j\) are reliability and cost of a unit to be used in the \(j\)-th subsystem.

The design data for this problem is shown in Table 1. The optimal solutions of the T-1 were already known as \(x = (3,4,5,3,3,2,4,5,4,3,4,3,4,5,5,5)\) with optimal value \(f(x) = 0.9456\).

5.2 Test Problem 2 (T-2)  This problem is an optimal redundancy allocation problem in the 10 component complex system in which \(x_j\) components at stage \(j\) are in parallel \((10)\). Fig. 4 shows such a complex system which is formulated mathematically as follows:
Table 1. Constant coefficients for T-1

<table>
<thead>
<tr>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_j</td>
<td>90</td>
<td>.75</td>
<td>65</td>
<td>.80</td>
<td>85</td>
<td>93</td>
<td>.78</td>
<td>66</td>
<td>.78</td>
<td>91</td>
<td>.79</td>
</tr>
<tr>
<td>c_j</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>w_j</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>j</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_j</td>
<td>.77</td>
<td>.67</td>
<td>.79</td>
<td>.67</td>
</tr>
<tr>
<td>c_j</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>w_j</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

Fig. 4. Complex system for T-2

\[
\begin{align*}
\text{max } f(x) &= [R_1(x_1) \cdot R_2(x_2) \cdot \Psi_1(x) + R_3(x_3) \cdot R_4(x_4) \cdot \Psi_2(x) + R_5(x_5) \cdot R_6(x_6) \cdot \Psi_3(x) + R_7(x_7) \cdot R_8(x_8) \cdot \Psi_4(x)] \cdot \Psi_5(x_9) \\
\text{s.t. } g_1(x) &= \sum_{j=1}^{10} (c_j \cdot x_j) \leq 125 \\
g_2(x) &= \sum_{j=1}^{10} (w_j \cdot x_j) \leq 10000 \\
x_j^L \leq x_j \leq x_j^U : \text{integer } j=1,\cdots,10
\end{align*}
\]

where

\[
\begin{align*}
\Psi_1(x) &= \prod_{j=1,2,3} R_j(x_j) + \prod_{j=4,7} R_j(x_j) + \prod_{j=5,6} R_j(x_j) + \prod_{j=8,7} R_j(x_j) - \prod_{j=1,2,3} R_j(x_j) - \prod_{j=4,7} R_j(x_j) - \prod_{j=5,6} R_j(x_j) - \prod_{j=8,7} R_j(x_j) \\
\Psi_2(x) &= \prod_{j=1,2,3} R_j(x_j) + \prod_{j=4,7} R_j(x_j) + \prod_{j=5,6} R_j(x_j) - \prod_{j=8,7} R_j(x_j) - \prod_{j=1,2,3} R_j(x_j) - \prod_{j=4,7} R_j(x_j) - \prod_{j=5,6} R_j(x_j) - \prod_{j=8,7} R_j(x_j) \\
\Psi_3(x) &= \prod_{j=1,2,3} R_j(x_j) + \prod_{j=4,7} R_j(x_j) + \prod_{j=5,6} R_j(x_j) - \prod_{j=8,7} R_j(x_j) - \prod_{j=1,2,3} R_j(x_j) - \prod_{j=4,7} R_j(x_j) - \prod_{j=5,6} R_j(x_j) - \prod_{j=8,7} R_j(x_j) \\
\Psi_4(x) &= \prod_{j=1,2,3} R_j(x_j) + \prod_{j=4,7} R_j(x_j) + \prod_{j=5,6} R_j(x_j) - \prod_{j=8,7} R_j(x_j) - \prod_{j=1,2,3} R_j(x_j) - \prod_{j=4,7} R_j(x_j) - \prod_{j=5,6} R_j(x_j) - \prod_{j=8,7} R_j(x_j) \\
R_j(x_j) &= 1 - R_j(x_j) \quad j=1,2,\cdots,10
\end{align*}
\]

Table 2. Constant coefficients for T-2

<table>
<thead>
<tr>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_j</td>
<td>.75</td>
<td>.65</td>
<td>.78</td>
<td>.85</td>
<td>.80</td>
<td>.75</td>
<td>.70</td>
<td>.70</td>
<td>.70</td>
<td>.69</td>
</tr>
<tr>
<td>c_j</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>w_j</td>
<td>250</td>
<td>110</td>
<td>350</td>
<td>720</td>
<td>690</td>
<td>140</td>
<td>300</td>
<td>450</td>
<td>200</td>
<td>350</td>
</tr>
<tr>
<td>x_j^L</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>x_j^U</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3. Computational results for T-1

<table>
<thead>
<tr>
<th>GA</th>
<th>na-GLS</th>
<th>a-GLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFV</td>
<td>0.9302</td>
<td>0.9456</td>
</tr>
<tr>
<td>AFV</td>
<td>0.9130</td>
<td>0.9455</td>
</tr>
<tr>
<td>CPU (sec.)</td>
<td>3.0958</td>
<td>0.9086</td>
</tr>
<tr>
<td>NGS</td>
<td>20</td>
<td>2</td>
</tr>
</tbody>
</table>

\( f(x) \) is system reliability, \( g_1(x) \) is total cost for the system, \( x_j \) is the decision variable which is the number of units in the \( j \)-th subsystem, \( g_2(x) \) is total physical limitation for the system. \( r_j \) and \( c_j \) are reliability and cost of a unit to be used in \( j \)-th subsystem, \( w_j \) is physical limitation( size or weight, etc) of the \( j \)-th subsystem.

Table 2 presents the design data for this problem. The optimal solution of T-2 is known as the objective value is

\[ f(x) = 0.9966 \quad \text{with } x=[1 2 1 1 1 5 5 6 6], \]

\[ g_1(x) = 103, \text{ and } g_2(x) = 9980. \]

These two examples (T-1 and T-2) are implemented in Visual Basic under IBM-PC Pentium 4 computer with 800 Mhz CPU speed and 526 MB RAM. The computational results using these two problems are appeared in Tables 3 and 4.

In Table 3, "BFV," "AFV," "CPU" and "NGS" mean the best fitness value, average fitness value, average CPU time and number of getting stuck at a local optimum after total iteration number is proceed, respectively.

In terms of the BFV, both the na-GLS and a-GLS located the optimal solution, but the CGA does not locate the optimal solution. In the AFV, the two hybrid algorithms (na-GLS and a-GLS) have same result and its result is better than that of the CGA. This implies that the iterative hill climbing method used in the na-GLS and a-GLS is well control the search to the optimal solution. In comparison of the CPU, the proposed a-GLS is significantly quicker than the CGA and na-GLS, which means that the adaptive scheme used in the a-GLS makes its search speed reduce by adaptively regulating the use of the local search. In terms of the NGS, the performance of the a-GLS is slightly superior to that of the na-GLS. This implies that the adaptive scheme used in the a-GLS is more efficient than the local search scheme in the na-GLS that does not use any adaptive scheme.

In Table 4, both the na-GLS and a-GLS have same results in terms of the BFV and AFV, which is better than those of the CGA. Especially, in the NGS, the CGA does not locate the optimal solution, but, the na-GLS and a-GLS always locate the optimal solution, in all trials. This means that the iterative hill climbing methods used in the na-GLS and a-GLS are efficient in locating the optimal solution. In comparison of the CPU, the a-GLS is the quickest, and the CGA is the slowest. This is explained by the
same reason with the analysis using the result of the CPU in Table 3.

For more various comparisons between the na-GLS and a-GLS, the number of the use of local search (NLS) is also analyzed in Fig. 5.

In Fig. 5 of T-2, the NLS of the na-GLS is significantly higher than that of the a-GLS. Similar result is also shown in T-2. This is implies that the adaptive scheme used in the a-GLS can reduce the number of the use of the local search significantly, which also can reduce the search time to the optimal solution rather than the na-GLS does. The comparisons of the CPU in Tables 3 and 4 explain this result.

6. Conclusion

In this paper, a new hybrid GA (a-GLS) for effectively solving reliability optimization problems is proposed. For the proposed algorithm, a new scheme for adaptively regulating the use of the local search in GA has been designed. The adaptive scheme measures the average fitness values from continuous two generations in GA. The proposed algorithm uses the iterative hill climbing method for local search.

For various comparisons, we have also suggested the canonical GA and hybrid GA (na-GLS) without any local search technique. The results by using two complex reliability optimization problems have shown the proposed a-GLS with adaptive scheme for local search is more efficient than the CGA and na-GLS without the scheme.

(Manuscript received Dec. 22, 2003, revised June 3, 2004)

References


Minoru Mukuda (Member) was born in Tokyo, Japan, on February 25, 1948. He is currently a Assoc Professor of Computer and Information Engineering, Nippon Institute of Technology, Japan. His major research filed is design of hybrid genetic algorithm, Fuzzy Logic, System Reliability Design, etc and Development of handicapped person help system. He has authored Hajime no C (fourth revision edition), Gijutsu-Hyohon Co., Ltd., Tokyo (2002).

YoungSu Yun (Non-member) was born in Pohang, Korea, on February 2, 1969. He received a Ph. D. degree in Industrial engineering from Konkuk University in 1998, and is presently a visiting researcher at Advanced Research Institute for Science and Engineering (RISE), Waseda University. His major research filed is design of hybrid genetic algorithm, adaptive algorithm, intelligent system, and so on.

Mitsuo Gen (Member) received the Ph.D. degree from the Kogakuin University, Japan in 1974. He was Lecturer at Ashikaga Institute of Technology, Japan in 1974-1980; Assoc Professor at Ashikaga Institute of Technology, Japan in 1982-1987, Professor at Ashikaga Institute of Technology, Japan in 1987-2003; He is currently a Professor of Graduate School of Information, Production & Systems, Waseda University. He was Visiting Assoc. Prof. at University of Nebraska Lincoln, USA in 1982-1983, Visiting Prof. at University of California at Berkeley, USA in 1999-2000. His research interests include Genetic Algorithms, Neural Network, Fuzzy Logic, and the applications of evolutionary techniques to Network Design, Schedule, System Reliability Design, etc. He has authored Genetic Algorithms & Engineering Design, John Wiley & Sons, New York (1997), Genetic Algorithms & Engineering Optimization, John Wiley & Sons, New York (2000) with Dr. Runwei Cheng, and is an Area Editor Journal of Computers & Industrial Engineering.