A Hybrid SS-SA Approach for Solving Multi-Objective Optimization Problems

Ramadan A. ZeinEldin
Operations Research department, Institute of Statistical Studies ad Research, Cairo University
E-mail: rzeanedean@yahoo.com

Abstract

Decision makers, nowadays, face complex real world problems having more than one conflicting objective functions to be optimized at the same time. In this paper, we developed a hybrid approach based on scatter search and simulated annealing for solving the multi-objective optimization problems. To validate our approach, we solved some test problems from the literature, compared the results with other approaches, and found that our proposed approach performs well.

Keywords: Multi-objective, optimization, Scatter search, simulated annealing.

1. Introduction

Multi-objective optimization is an important research topic for scientists and researchers. This is due to the multi-objective nature of real world problems. Most real problems are complex and multidisciplinary in nature, and quite often require more than one conflicting objective functions to be optimized simultaneously while usually no prior information of their exact interactions is available.

Researchers have developed many multi-objective optimization procedures. For multi-objective optimization problems, there is not a single optimum solution, but a set of non-dominated optimal solutions called the Pareto set of solutions. The challenge is in the case of conflicting objectives, which is usually the case in most real problems.

Traditional mathematical programming techniques have some limitations when solving MOPs. Most of them depend on the shape of the Pareto front and only generate one Pareto solution from each run. Thus, several runs (with different parameter settings) are generally required to generate a Pareto solution set; however, sometimes different parameter settings may generate similar results. In such circumstances, generating a Pareto solution set will be very computationally expensive [Li and Du, 2013]. In the last two decades most of the researchers used meta-heuristics approaches such as genetic algorithm, simulated annealing, tabu search, particle swarm to solve the multi-objective problems.

Simulated annealing algorithm repeats an iterative neighbor generation procedure and follows search directions that improve the objective function value. While exploring solution space, the SA method offers the possibility of accepting worse neighbor solutions in a controlled manner in order to escape from local minima.

In terms of metaheuristics, recently, scatter search approaches are receiving increasing attention, because of their potential to effectively explore a wide range of complex optimization problems [Silva et al. 2013]. Hybrid algorithm can make good use of the characteristics of different algorithms to achieve complementary advantages to improve the algorithm optimal performance and efficiency [Zhang et al. 2012].
In this paper, a hybrid approach based on scatter search and simulated annealing is developed to solve the multi-objective problems. The remainder of this article is organized as follows. The problem definition and the literature review for the multi-objective optimization problem are presented in section 2. Simulated annealing is presented in section 3 and scatter search is presented in section 4. The proposed hybrid approach is presented in section 5, and section 6 is assigned to the implementation of the proposed approach. Finally, the conclusions and points for future research are mentioned in section 7.

2. Multi Objectives and Literature Review

Multi-objective optimization problems contain more than one objective that needs to be achieved simultaneously. Such problems arise in many applications, where two or more, sometimes competing and/or incommensurable objective functions have to be minimized concurrently. This group of optimization problems has a rather different perspective compared to single objective problems. In the single objective optimization there is only one global optimum, but in multi-objective optimization there is a set of solutions, called the Pareto-optimal set, which are considered to be equally important; all of them constitute global optimum solutions (Kaveh and Laknejadi, 2011). Multi-objective optimization requires more computational effort than single-objective optimization. Unless preferences are irrelevant or completely understood, solution of several single objective problems may be necessary to obtain an acceptable final solution.

The Multi-Objective Optimization Problem is a vector optimization problem stated as follows: (Marler and Arora, 2004; Kaveh and Laknejadi, 2011; Bajestani et al., 2009)

Minimize: \( F(X) = \{f_1(x), f_2(x), \ldots, f_p(x)\} \)

Subject to: \( g_j(x) \leq 0, \ j = 1, m \) inequality constraints
\( h_k(x) = 0, \ k = 1, l \) equality constraints
\( x^l_i \leq x_i \leq x^u_i, i = 1, n \) side constraints

Most multi-objective-programming techniques focus on finding the set of efficient points for a given problem or, in the case of heuristic procedures, an approximation of the efficient set (Molina et al. 2007). The two critical challenges in MOP are: 1) The generation of a Pareto set of solutions; and 2) the selection of a preferred final design from the Pareto set of solutions generated. Multi-objective optimization solution strategies have been classified into the following categories, based on how selecting a preference setting is implemented: 1) no preference method; 2) the a priori method, – where a decision maker's preferences are taken into consideration before the optimization; 3) the a posteriori method, – where a design is selected based on preferences after the Pareto set is generated; and 4) interactive methods, – where the decision maker can incorporate and change preferences during the solution process (Miettinen, 1999). In the last decade, evolutionary approaches have been the primary tools to solve real-world multi-objective problems (Konak et al., 2006).


3. An Overview about Simulated Annealing

Simulated annealing (SA) is a robust technique, which provides excellent solutions to single and multiple objective optimization problems with a substantial reduction in computation time. Suman and Kumar (2006) developed a comprehensive review of simulated annealing based optimization algorithms for solving single and multi-objective optimization problems. SA has shown to be effective in finding the optimal solution for both discrete and continuous optimization problems (Xhafa et al., 2011).


The SA algorithm consists of four main components (Jolai et al., 2013):
(i) Configurations;
(ii) Re-configuration technique;
(iii) Cost function;
(iv) Cooling schedule.

SA uses a stochastic approach to direct the search. It allows the search to proceed to a neighboring state even if the move causes the value of the objective function to become worse. It guides the original local search method in the following way. If a move to a neighbor $X'$ in the neighborhood $N(X)$ decreases the objective function value, or leaves it unchanged, then the move is always accepted. More precisely, the solution $X'$ is accepted as the new current solution if $\Delta \leq 0$, where $\Delta = C(X') - C(X)$ and $C(X)$ is the value of the objective function. Moves, which increase the objective function value, are accepted with a probability of $e^{\Delta/T}$ to allow the search to escape a local optimum, where $T$ is a parameter named as temperature. The value of $T$ varies from a relatively large value to a small value close to zero. These values are controlled by a cooling schedule that specifies the initial
and incremental temperature values at each stage of the algorithm. The details of SA algorithm is mentioned in (Tavakkoli-Moghaddam et al., 2011; Abbasi et al., 2011; Ribeiro et al., 2011; Xhafa et al., 2011).

4. An Overview about Scatter Search

Scatter search (SS) is an evolutionary method that has been successfully applied to hard optimization problems. The fundamental concepts and principles of the method were first proposed in the 1970s, based on formulations dating back to the 1960s to combine decision rules and problem constraints. In contrast to other evolutionary methods like genetic algorithms, scatter search is founded on the premise that systematic designs and methods for creating new solutions afford significant benefits beyond those derived from recourse to randomization. It uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems (Marti et al., 2006). Scatter search is a population based meta-heuristic method that uses a reference set to combine its solutions and construct others. Scatter search is an evolutionary method that has been successfully applied to hard optimization problems. The key idea of SS in multi-objective optimization is to maintain a good search time to balance the solutions’ concentration and diversity so that the solutions on the Pareto border has certain dispersion (Zhang et al., 2012). Diaz et al. (2006) performed an empirical evaluation of the effectiveness of different ways of parallelizing the SS algorithm. They identified two sets of tasks within the SS template that can be parallelized, and to design several simple, intuitive ways of carrying out this parallelization. Gortazar et al. (2010) adapted SS to develop a black box solver for optimization problems with binary variables. Herrera et al. (2006) deal with a continuous version of the scatter search, which works directly with vectors of real components. Ibaneez et al. (2012) proposed a new skull-face overlay method based on the scatter search evolutionary algorithm. The SS framework has been widely studied because it is very flexible and can be easily applied to many diverse optimization problems. For instance, some steps of the SS framework can be omitted or the procedures can be rearranged. In the recent years, the SS framework has been widely used to solve many complex combinatorial optimization problems, and it was shown to be more effective than other metaheuristics in many instances (Zhang et al., 2012). Marti (2006) reviewed the scatter search implementations and developments from 1994 to 2006. He also showed the cumulative frequency, which shows the dramatic increase in the number of publications dealing with SS. From these applications: methodology, assignment, binary problems, clustering/selection, coloring, commercial soft, continuous, graph drawing, graph problems, knapsack, linear ordering, mixed integer programming, multiobjective, neural networks, p-Median, permutation problems, routing, scheduling, and software testing. Marti (2002) proposed a SS to solve a real-life problem with multiple objectives. Pendharkar (2013) presented an interactive multi-criteria procedure that combine scatter and random search for coal production planning problem with fuzzy profit and fuzzy coal quality decision-maker utilities. Silva et al. (2013) proposed an improved scatter search to deal with multiobjective environmental/economic dispatch problems based on concepts of Pareto dominance and crowding distance and a new scheme for the combination method.

A scatter search implementation consists of the following five methods (Marti et al., 2006):

1) A diversification generation method to generate a collection of diverse trial solutions, using an arbitrary trial solution (or seed solution) as an input.

2) An improvement method to transform a trial solution into one or more enhanced trial solutions. (Neither the input nor the output solutions are required to be feasible, though the output solutions will more usually be expected to be so. If no improvement of the input trial solution results, the “enhanced” solution is considered to be the same as the input solution.)

3) Reference set update method to build and maintain a reference set consisting of the b “best” solutions found (where the value of b is typically small, e.g., no more than 20),
organized to provide efficient accessing by other parts of the method. Solutions gain membership to the reference set according to their quality or diversity.

4) A subset generation method to operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions.

5) Solution combination method to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solution vectors.

5. The Proposed Hybrid SS-SA Approach
In this section we describe the proposed approach SS-SA.

5.1. The Diversification Generation Method
This method is applied to generate diverse solutions. In the proposed hybrid approach, the initial population is generated using the diversification generation method. The weighted method is used to generate diverse solutions. The weighted method generates 100 solutions; each solution generated is checked for feasibility although the SS accept infeasible solutions in the first phase but we check feasibility at the first phase to save time in the coming phases. Diversification generation method can construct initial trial solution set quickly. Most SS algorithms generally use heuristics to obtain initial trial solutions and adopt methods with few runtime to improve solutions. The procedure of generating new solutions is usually developed according to the characteristic of problem, which is the essential of an SS algorithm. However, generating the initial trial solutions and the methods of improving solution are important in SS (Tang et al., 2010).

5.2. The Improvement Method
The improvement method is generally used after generating new solutions. The new solutions are the initial trial solutions or solutions generated by combination method. Hence, the improvement method need treat with feasible and infeasible solutions. But in our proposed hybrid approach we check feasibility and select only the feasible solution to save time and find out quality solutions. In our proposed SS-SA, the SA approach is used to improve the previously generated solutions. And from the implementation we found that the SA leads to better solutions.

5.3. The Reference Set Update Method
The scatter search algorithm constructs and updates the refset using this method. The reference set update method stores the solutions in the refset according to their quality to form the subset refset1 and according to their diversity, to form the subset refset2. In order to maintain appropriate diversity in refset1 the SA is used. The solutions in the refset1 are sorted according to the values of the objective functions (the minimum values). The SA steps are applied on the best solution of the refset1 to generate other good solutions. The good generated solutions replace the bad ones. As known the SA algorithm jumps from local minimum to global minimum.

After updating refset, if there is no difference between new refset and the previous one, the refset rebuilding approach is used. In this approach, refset2 is cleared and filled with generated solutions using the diversification generation method.

Reference set update method completes building the initial reference set and updating the reference set when the new solution is created.

5.4. The Subset Generation Method
Before the combination method is used to generate the new solutions, the subset generation method forms subsets which are used in the combination method. The proposed subset generation method
divides refset solutions into three kinds of desired subsets. Each of these subsets consists of one or two solutions. In our implementation, we use 2-element subsets as a type of subsets.

5.5. The Combination Method

In order to generate new solutions, two methods are used; the SA method and two-point crossover. We check feasibility before selecting the new solution. We found that the solutions generated using SA are better than the second method.

6. Computational Analysis

To evaluate our proposed approach, test problems are used and solved using the proposed approach. These test problems are solved in the literature so we compare the results of our proposed approach with the results of the other approaches used these problems. The proposed hybrid approach is coded using Matlab version R2010a and executed on Intel(R) Core(TM) i3, 2.13 GHz, Windows 7 using 2 GB of RAM.

6.1. The Test Problems

These test problems are used in most of the research in the literature such as Caponio et al. (2007); Zou et al. (2013); Yang et al. (2013); Enayatifar (2013); Gong et al. (2013); Jiao et al. (2013); Kaveh and Laknejadi (2011); Beausoleil (2006); Deb et al. (2002). Table 1 shows the test problems.

<table>
<thead>
<tr>
<th>Prob.</th>
<th>N</th>
<th>Bounds</th>
<th>Objective functions</th>
<th>Solutions</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZDT1</td>
<td>30</td>
<td>[0, 1]</td>
<td>$f_1(x) = x_1^4$</td>
<td>$x_1 \in [0, 1]$</td>
<td>Convex</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$f_2(x) = g(x) \left( 1 - \frac{x_1}{\sqrt{g(x)}} \right)$</td>
<td>$x_i = 0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$g(x) = 1 + 9 \sum_{i=2}^{n} \frac{x_i^2}{n-1}$</td>
<td>$i = 2, \ldots, n$</td>
<td></td>
</tr>
<tr>
<td>ZDT2</td>
<td>30</td>
<td>[0, 1]</td>
<td>$f_1(x) = x_1$</td>
<td>$x_1 \in [0, 1]$</td>
<td>Nonconvex</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$f_2(x) = g(x) \left( 1 - \left( \frac{x_1}{g(x)} \right)^2 \right)$</td>
<td>$x_i = 0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$g(x) = 1 + 9 \sum_{i=2}^{n} \frac{x_i^2}{n-1}$</td>
<td>$i = 2, \ldots, n$</td>
<td></td>
</tr>
<tr>
<td>ZDT3</td>
<td>30</td>
<td>[0, 1]</td>
<td>$f_1(x) = x_1$</td>
<td>$x_1 \in [0, 1]$</td>
<td>Convex, disconnected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$f_2(x) = g(x) \left( 1 - \frac{x_1}{\sqrt{g(x)}} \right.$</td>
<td>$x_i = 0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$- \frac{x_1}{g(x)} \sin(10\pi x_1) \right)$</td>
<td>$i = 2, \ldots, n$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$g(x) = 1 + 9 \sum_{i=2}^{n} \frac{x_i^2}{n-1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZDT4</td>
<td>10</td>
<td>$x_1 \in [0, 1]$</td>
<td>$x_i \in [-5, 5]$</td>
<td>$x_1 \in [0, 1]$</td>
<td>Nonconvex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$i = 2, \ldots, n$</td>
<td>$x_i = 0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$f_2(x) = g(x) \left( 1 - \frac{x_1}{\sqrt{g(x)}} \right)$</td>
<td>$i = 2, \ldots, n$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$g(x) = 1 + 10(n-1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$+ \sum_{i=2}^{n} \left( x_i^2 - 10\cos(4\pi x_i) \right)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 1: Test problems - continued

| ZDT6 | 10 | [0, 1] | \( f_1(x) = 1 \) - \( \exp(-4x_1) \sin^6(4\pi x_1) \) | \( f_2(x) = g(x) \left[ 1 - \left( \frac{x_1}{g(x)} \right)^2 \right] \) | \( x_1 \in [0, 1] \) | \( x_1 = 0 \) | \( i = 2, \ldots, n \) | Nonconvex, nonuniformly spread |

6.2. Comparison Metrics

It is difficult to compare results of one multi-objective method to another, as there is not a unique optimum in multi-objective optimization as in single objective optimization. So, the best solution in multi-objective terms is decided by the decision maker (Suman and Kumar, 2006). So there are some comparison metrics that are used to compare among developed approaches. Rahimi-Vahed et al., (2008) mentioned five comparison metrics. They are the number of Pareto solutions, error ratio, generational distance, spacing metric and diversification metric. Where:

1. **The number of Pareto solutions.** This metric shows the number of Pareto optimal solutions that each algorithm can find. The number of found Pareto solutions corresponding to each algorithm is compared with the total Pareto-optimal solutions obtained by the total enumeration algorithm.

2. **Error ratio.** This metric allows us to measure the non-convergence of the algorithms towards the Pareto-optimal frontier. The error ratio is given by:

\[
E = \frac{\sum_{i=1}^{N} e_i}{N}
\]

Where \( N \) is the number of found Pareto-optimal solutions, and \( e_i \) is zero if the solution \( i \in \) Pareto optimal frontier and 1 otherwise. The closer this metric is to 1, the less the solution has converged toward the Pareto-optimal frontier.

3. **Generational distance (GD).** This metric allows measurement of the distance between the Pareto-optimal frontier and the solution set. The definition of this metric is:

\[
GD = \frac{\sum_{i=1}^{N} d_i}{N}
\]

Where \( d_i \) is the Euclidean distance between solution \( i \) and the closest which belongs to the Pareto-optimal frontier obtained from the total enumeration.

4. **Spacing metric (S).** The spacing metric gives a measure of the uniformity of the spread of points of the solution set. The metric of spacing [233] gives an indication of how evenly the solutions are distributed along the discovered front.

It is given by:

\[
S = \left[ \frac{1}{N - 1} \sum_{i=1}^{N} (d - d_i)^2 \right]^{1/2}
\]

Where \( d \) is the mean value of all \( d_i \).

5. **Diversification metric (D).** This metric measures the spread of the solution set and is defined as:

\[
D = \sqrt{\sum_{i=1}^{N} \max(||x'_i - \hat{y}_i||)}
\]

Where \( ||x'_i - \hat{y}_i|| \) is the Euclidean distance between the non-dominated solution \( x'_i \) and the nondominated solution \( \hat{y}_i \).
6.3. Parameter Setting

We have run the proposed approach with different values for the parameters of SS and SA and we reached to the following parameters values to get quality solutions:

- The population size is set to 100 solutions
- The maximum size of RefSet1, $b_1$, is set to 30 and the maximum size of RefSet2, $b_2$, is set to 30.
- $T_0 = 0.95$, $\alpha = 0.95$, $T_f = 0.05$, N (number of iterations at each temperature) = 20

6.4. Results Comparisons

We solved all the test problems and the following tables show the performance comparison between our proposed hybrid approach and other approaches mentioned in (Kaveh and Laknejadi, 2011) for some of the problems stated in table 1. Table 2 shows the results for the problem ZDT1, Table 3 shows the results for the problem ZDT3, Table 4 shows the results for the problem ZDT4, Table 5 shows the results for the problem ZDT6. These tables show the mean value, standard deviation (s) and the time required per run for the metrics mentioned in section 5.2. From the comparison, we see that our proposed approach perform well for most of the problems. For some problems it gives results near to the other approaches compared with.

Table 2: Results for problem ZDT1

<table>
<thead>
<tr>
<th></th>
<th>NSG-II</th>
<th>PEAS</th>
<th>MOPSO</th>
<th>CSS-MOPSO</th>
<th>SS-SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD (mean)</td>
<td>0.003731</td>
<td>0.004932</td>
<td>0.096400</td>
<td>0.003048</td>
<td>0.004230</td>
</tr>
<tr>
<td>GD (s)</td>
<td>0.000342</td>
<td>0.006013</td>
<td>0.011428</td>
<td>0.000288</td>
<td>0.000372</td>
</tr>
<tr>
<td>S (mean)</td>
<td>0.503569</td>
<td>3.765871</td>
<td>0.756200</td>
<td>0.199631</td>
<td>0.216251</td>
</tr>
<tr>
<td>S (s)</td>
<td>0.052127</td>
<td>1.367000</td>
<td>0.145703</td>
<td>0.048466</td>
<td>0.061742</td>
</tr>
<tr>
<td>D (Mean)</td>
<td>1</td>
<td>0.901099</td>
<td>0.725329</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D (s)</td>
<td>0</td>
<td>0.077259</td>
<td>0.020220</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Time (min)</td>
<td>0.551</td>
<td>0.029</td>
<td>0.091</td>
<td>0.098</td>
<td>0.123</td>
</tr>
</tbody>
</table>

Table 3: Results for problem ZDT3

<table>
<thead>
<tr>
<th></th>
<th>NSG-II</th>
<th>PEAS</th>
<th>MOPSO</th>
<th>CSS-MOPSO</th>
<th>SS-SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD (mean)</td>
<td>0.005031</td>
<td>0.082004</td>
<td>0.068005</td>
<td>0.004781</td>
<td>0.00523</td>
</tr>
<tr>
<td>GD (s)</td>
<td>0.000162</td>
<td>0.107889</td>
<td>0.013964</td>
<td>0.000141</td>
<td>0.000197</td>
</tr>
<tr>
<td>S (mean)</td>
<td>0.502427</td>
<td>2.044142</td>
<td>0.794325</td>
<td>0.301347</td>
<td>0.414025</td>
</tr>
<tr>
<td>S (s)</td>
<td>0.047587</td>
<td>1.500228</td>
<td>0.070546</td>
<td>0.042321</td>
<td>0.050318</td>
</tr>
<tr>
<td>D (Mean)</td>
<td>0.929290</td>
<td>0.583419</td>
<td>0.695206</td>
<td>0.929012</td>
<td>0.825601</td>
</tr>
<tr>
<td>D (s)</td>
<td>0.000438</td>
<td>0.177700</td>
<td>0.048935</td>
<td>0.000846</td>
<td>0.007034</td>
</tr>
<tr>
<td>Time (min)</td>
<td>1.019</td>
<td>0.052</td>
<td>0.213</td>
<td>0.216</td>
<td>0.411</td>
</tr>
</tbody>
</table>

Table 4: Results for problem ZDT4

<table>
<thead>
<tr>
<th></th>
<th>NSG-II</th>
<th>PEAS</th>
<th>MOPSO</th>
<th>CSS-MOPSO</th>
<th>SS-SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD (mean)</td>
<td>0.003559</td>
<td>0.450385</td>
<td>0.228776</td>
<td>0.003462</td>
<td>0.00169</td>
</tr>
<tr>
<td>GD (s)</td>
<td>0.000589</td>
<td>0.063752</td>
<td>0.070243</td>
<td>0.000380</td>
<td>0.003215</td>
</tr>
</tbody>
</table>

Table 4: Results for problem ZDT4 - continued

<table>
<thead>
<tr>
<th></th>
<th>NSG-II</th>
<th>PEAS</th>
<th>MOPSO</th>
<th>CSS-MOPSO</th>
<th>SS-SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>S (mean)</td>
<td>0.485384</td>
<td>2.799198</td>
<td>1.081437</td>
<td>0.175199</td>
<td>0.29452</td>
</tr>
<tr>
<td>S (s)</td>
<td>0.052186</td>
<td>2.015591</td>
<td>0.195925</td>
<td>0.031783</td>
<td>0.04320</td>
</tr>
<tr>
<td>D (Mean)</td>
<td>1</td>
<td>13.36665</td>
<td>3.430176</td>
<td>0.999987</td>
<td>2.06781</td>
</tr>
<tr>
<td>D (s)</td>
<td>0</td>
<td>3.921435</td>
<td>2.176589</td>
<td>0.000071</td>
<td>0.50673</td>
</tr>
<tr>
<td>Time (min)</td>
<td>1.282</td>
<td>0.049</td>
<td>0.251</td>
<td>0.213</td>
<td>0.518</td>
</tr>
</tbody>
</table>
Table 5: Results for problem ZDT6

<table>
<thead>
<tr>
<th></th>
<th>NSG-II</th>
<th>PEAS</th>
<th>MOPSO</th>
<th>CSS-MOPSO</th>
<th>SS-SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD (mean)</td>
<td>0.024666</td>
<td>0.01739</td>
<td>0.019078</td>
<td>0.026345</td>
<td>0.030612</td>
</tr>
<tr>
<td>GD (s)</td>
<td>0.024338</td>
<td>0.016882</td>
<td>0.009291</td>
<td>0.014011</td>
<td>0.019813</td>
</tr>
<tr>
<td>S (mean)</td>
<td>2.151897</td>
<td>3.499715</td>
<td>3.707637</td>
<td>3.179233</td>
<td>4.017011</td>
</tr>
<tr>
<td>S (s)</td>
<td>2.285011</td>
<td>2.217825</td>
<td>0.849501</td>
<td>1.351519</td>
<td>2.001026</td>
</tr>
<tr>
<td>D (Mean)</td>
<td>1</td>
<td>0.933755</td>
<td>0.903402</td>
<td>1</td>
<td>0.986012</td>
</tr>
<tr>
<td>D (s)</td>
<td>0</td>
<td>0.126736</td>
<td>0.267582</td>
<td>0</td>
<td>0.193544</td>
</tr>
<tr>
<td>Time (min)</td>
<td>1.032</td>
<td>0.037</td>
<td>0.101</td>
<td>0.119</td>
<td>0.986</td>
</tr>
</tbody>
</table>

7. Conclusions

In this paper we have presented a hybrid approach based on scatter search and simulated annealing to solve the multi-objective optimization problems. Different test problems were used to compare the performance of our approach with other approaches. The results show that our proposed approach is effective and competitive with the other developed approaches in the literature.

Our recommendations are to develop studies for determining the best values for the SS and SA parameters, dealing with dynamic multi-objective optimization problems and stochastic multi-objective optimization problems.

References


A Hybrid SS-SA Approach for Solving Multi-Objective Optimization Problems


