CONTINUOUS OPTIMIZATION PROBLEM SOLUTION WITH SIMULATED ANNEALING AND GENETIC ALGORITHM

Pakize ERDOĞMUŞ*, Ali ÖZTÜRK*, Salih TOSUN

Düzce Üniversitesi, Teknik Eğitim Fakültesi, Beştevi, Düzce, TÜRKİYE, E-mail: pakizeerdogmus@duzce.edu.tr
Düzce Üniversitesi, Teknik Eğitim Fakültesi, Elektrik Eğitim, Beştevi,Düzce,TÜRKİYE, E-mail: aliotsurk@duzce.edu.tr, salihitosun@duzce.edu.tr

Abstract

Simulated Annealing and Genetic Algorithm are two well-known metaheuristic algorithms for combinatorial optimization. These two methods have also been used for solving constrained continuous problems. In this study, five constrained continuous problems have been solved both Simulated Annealing (SA) and Genetic Algorithm (GA). Optimum results have been compared with real optimum values obtained with continuous optimization methods. It has been seen that combinatorial optimization methods can successfully be applied to continuous optimization problems.

Keywords: Linear Programming, Non-linear Programming, Genetic Algorithm, Simulated Annealing

1. Introduction

In last decades there has been a great deal of interest on the applications of heuristic search algorithms to the continuous problems [1, 2, 3, 9, 10]. These metaheuristics have been applied to a lot of optimization problems and it has been taken successful results with these metaheuristics[10,11,12]. It is a fact that a variety of applications in engineering, decision science, and operations research have been formulated as constrained continuous optimization problems. Such applications include neural-network learning, digital signal and image processing, structural optimization, engineering design, computer-aided-design (CAD) for VLSI, database design and processing, nuclear power plant design and operation, mechanical design, and chemical process control. Optimal or good solutions to these applications have significant impacts on system performance, such as low-cost implementation and maintenance, fast execution, and robust operation [3].

Constrained global optimization is NP-hard [3], because it takes exponential time to verify whether a feasible solution is optimal or not for a general constrained problem. Solution methods of constrained continuous problems have been classified in three main categories. Analytic methods, Decomposition methods and Metaheuristics. Analytic methods can only solve some trivial cases of constrained problems, while decomposition methods are enumerative methods that decompose a large problem into smaller sub problems that are easier to solve. Metaheuristic methods sample search space based on some probability distributions. Depending on the coverage of search space, the best solution is found related to the performance of used algorithm [3].

In this paper, metaheuristics have been used for solving constrained continuous optimization problems. Since it has been aimed to both, to compare the performances of the two well-known metaheuristics and to show these metaheuristics good performance on the constrained continuous optimization problems, it has been used five test problems. Some of these problems have been taken from literature [2].

2. Constrained Continuous Problems

Both Linear and Non-linear programming problems are also called continuous problems. In this paper, we will focus on constrained continuous optimization problems including Linear and Non-linear problems.

The constrained continuous optimization problem is formulated as follows.

\[ \text{Opt}(\min \text{ or } \max) \ f(x) \quad (x \in \mathbb{R}) \]

Constrained to: \( g_i(x) < 0, \quad i = 1, \ldots, k \)
\( h_j(x) = 0, \quad j = 1, \ldots, l \)

If \( f(x), g(x) \) and \( h(x) \) are linear functions of variables, then it is called linear and if \( f(x) \) is not a linear function of variables then the problem is called non-linear.

And the problems used for testing GA and SA performance on continuous problems are as below.

Table 1. Test Problems for used to see SA and GA performance on Continuous Problems

<table>
<thead>
<tr>
<th>Problem no</th>
<th>Object Function</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.(Linear)</td>
<td>( f_{\text{max}}=3x_1+2x_2 )</td>
<td>( 2x_1+x_2&lt;50 ) ( x_1&lt;15 )</td>
</tr>
<tr>
<td>2.(Linear)</td>
<td>( f_{\text{max}}=5x_1+7x_2+8x_3 )</td>
<td>( x_1+2x_2&lt;150 ) ( 5x_2+3x_3&lt;160 )</td>
</tr>
<tr>
<td>3.(Non-linear)</td>
<td>( f_{\text{min}}=-x_1^2x_2^3 )</td>
<td>( 0 \leq x_1+x_2+2x_3&lt;72 ) %These constraint divide two part; ( -x_1-2x_2+2x_3&lt;0 ) ( x_1+2x_2+2x_3&lt;72 )</td>
</tr>
<tr>
<td>4.(Non-Linear)</td>
<td>( f_{\text{min}}=-2x_1-6x_2+x_1^2+8x_2^2 )</td>
<td>( x_1+6x_2&lt;6 ) ( 5x_1+4x_2&lt;10 ) ( 0 &lt; x_1 &lt; 2 ) ( 0 &lt; x_2 &lt; 1 )</td>
</tr>
<tr>
<td>5.(Non-linear)</td>
<td>( f_{\text{max}}=x_1^3+x_2^5(\text{Non-Linear}) )</td>
<td>( x_1^2+x_2&lt;2 ) ( x_1^2-x_2&lt;0 ) ( -3x_2&lt;0 ) ( 0 &lt; x_2 &lt; 5 )</td>
</tr>
</tbody>
</table>
3. Metaheuristics

Metaheuristics methods have been considered to be acceptably good solvers of unconstrained continuous problems [1]. The power of metaheuristic methods comes from the fact that they are robust and can deal successfully with a wide range of problem areas. However, when these methods are applied to complex problems, it has been seen their slow convergence. The main reason for this slow convergence is that these methods explore the global search space by creating random movements without using much local information. In contrast, local search methods have faster convergence due to their using local information to determine the most promising search direction by creating logical movements. However, local search methods can easily be entrapped in local minima [1]. Both SA and GA run according to unconstrained optimization procedure. The constrained continuous optimization problems have been transformed into unconstrained continuous optimization problems by penalizing the objective function value with the quadratic penalty function. In case of any violation of a constraint boundary, the fitness of corresponding solution is penalized, and thus kept within feasible regions of the design space by increasing the value of the objective function when constraint violations are encountered [14].

\[ P = \sum_{i=1}^{k} R \left( \max \{ 0, g_i \} \right)^2 + \sum_{j=1}^{l} R \left( \max \{ 0, h_j \} \right)^2 \]

By this penalty function, if constraints are in feasible region, then \( P \) is equal zero and if not the fitness function or objective function is penalized by \( P \).

4. Simulated Annealing and Genetic Algorithm

4.1. Simulated Annealing

Simulated Annealing (SA), is a global optimization algorithm inspired by physical annealing process of solids [4].

Annealing is cooled down slowly in order to keep the system of the melt in a thermodynamic equilibrium which will increase the size of its crystals and reduce their defects. As cooling proceeds, the atoms of solid become more ordered. The initial temperature must be high enough in order to avoid a local minimum of energy.

SA was originally based on statistical Metropolis algorithm.

SA aims to find global minimum without got trapped local minimums. So if object function is a maximization problem, problem is converted minimization problem multiplying minus 1. The algorithm of SA is as Figure 1.

Algorithm starts with an initial solution an initial temperature. Search process continuous while stopping criteria is true. Maximum run time, maximum iteration number, e.g. may be stopping criteria. For each \( T \), \( s' \in N(s) \) is selected randomly. And if \( f(s') < f(s) \) then \( s' \) is accepted new solution like local search. But if \( f(s') > f(s) \) then \( x = U(0,1) \) is produced and if \( x \) smaller \( P(s', s; T) \) then \( s' \) is also accepted as new solution for diversification.

\[ P = e^{-\frac{(f(s')-f(s))}{T}} \]

So firstly the possibility of bad solutions acceptance or(hill-climbing moves) is high for diversification.

\[ \lim_{T \to \infty} e^{-\frac{(f(s')-f(s))}{T}} = e^0 = 1 \]

\( T \) is decreased along the search process. The possibility of bad solution acceptance is approach to zero. And the process converges to local search method for intensification.

\[ \lim_{T \to 0} e^{-\frac{(f(s')-f(s))}{T}} = e^{-\infty} = 0 \]
The proper annealing process is related initial temperature, iteration for each temperature, temperature decrement coefficient and stopping criteria. All these criteria can be found related articles[5].

4.2. Genetic Algorithm

A genetic algorithm (GA) is a search technique aimed to find optimal solution. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover [6].

Simple generational genetic algorithm flowchart is as Figure 2.

![GA Algorithm Flowchart](image)

Figure 2. GA Algorithm

More detailed information about GA can be found[7,8].

5. Results

In this study it has been proven that metaheuristics can be used to solve continuous constraints problem solutions. As it has been seen in the Table 2, results are very near to real optimum value. So it can be advised metaheuristics, when the problem size is big or analytic solution method is difficult.

<table>
<thead>
<tr>
<th>Prob no</th>
<th>V of S</th>
<th>Real Optimum Function Value and variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fopt</td>
<td>X1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>82.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>668.333</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-3456</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-2.2137</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>19.8100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prob no</th>
<th>V of S</th>
<th>Fopt</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>82.4621</td>
<td>17.4890</td>
<td>14.9975</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>648.812</td>
<td>48.3146</td>
<td>0.04885</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-3434.1</td>
<td>25.2076</td>
<td>11.2848</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-2.21366</td>
<td>0.81680</td>
<td>0.37469</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>19.8070</td>
<td>-1.9838</td>
<td>3.9838</td>
<td></td>
</tr>
</tbody>
</table>

6. References


Erdogmus, P., Ozturk, A., Tosun, S. Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, Urbana, Illinois, 2001.


