Continuous optimization problem solution with simulated annealing and genetic algorithms

P. Erdogmus¹, A. Ozturk², S. Tosun³

¹Duzce University, Computer Engineering, Duzce, Turkey.
²,³Duzce University, Electrical and Electronics Engineering, Duzce, Turkey.

Accepted 24 Feb 2013

Abstract
Simulated Annealing (SA) and Genetic Algorithm (GA) 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 SA and 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 algorithm

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.

2. Constrained continuous problem
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.

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

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 Table 1.

3. Metaheuristics
Meta-heuristics methods have been considered to be acceptably good solvers of unconstrained continuous
The power of meta-heuristic 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 problem 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].

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>( 2x_1+x_2+x_3&lt;150 ) ( x_1+2x_2&lt;125 ) ( 5x_1+3x_2&lt;160 )</td>
</tr>
<tr>
<td>3.(Non-linear)</td>
<td>( f_{\text{min}}=-x_1x_2x_3 )</td>
<td>( 0&lt;x_1+2x_2+2x_3&lt;72 ) %These constraint divide two part; ( -x_1^2x_2^2-2x_1&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_1^3+8x_2^2 )</td>
<td>( x_1^2x_2&lt;6 ) ( 5x_1^2+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_1^2+x_2^2 )</td>
<td>( X_1+x_2&lt;2 ) ( X_1^2-x_2&lt;0 ) ( -3&lt;x_1&lt;2 ) ( 0&lt;x_2&lt;5 )</td>
</tr>
</tbody>
</table>

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 \( P \). This case is given by equation 1. Problem 5 fitness function for both GA and SA can be formulated given equation 2.

\[
P = \sum_{i=1}^{k} R_i (\max[0, h_i])^2 0 + \sum_{i=1}^{k} R_i (\max[0, h_i])^2
\]  

(1)

\[
F_{\text{fitness}} = x_1^2 + x_2^2 + R_1 \max(0, x_1 + x_2 - 2)^2 + R_2 \max(0, x_1^2 + x_2)
\]  

(2)

4. Simulated annealing and genetic algorithm

GA and SA are the most well-known and most used heuristic algorithms. When it has been researched in Science Direct with the given keywords as in Table 2 between 1970-2013, it has been found that GA and SA are most popular two optimization algorithms in the literature. GA is a method for finding both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. SA known thermal process for obtaining low energy state of a solid in a heat bath. This process consists of two steps. First, increase the temperature of the heat bath to a maximum value at which the solid melts. secondly,
decrease the temperature carefully until the particles arrange themselves in a ground state of the solid.

### Table 2. Science Direct research results

<table>
<thead>
<tr>
<th>Keyword</th>
<th>The number of articles</th>
<th>The birth date of algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genetic Algorithm</td>
<td>135,739</td>
<td>1976</td>
</tr>
<tr>
<td>Simulated Annealing</td>
<td>53,176</td>
<td>1983</td>
</tr>
<tr>
<td>Particle Swarm Optimization</td>
<td>9,556</td>
<td>1995</td>
</tr>
<tr>
<td>Artificial Bee Colony</td>
<td>3,306</td>
<td>2004</td>
</tr>
</tbody>
</table>

#### 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’ ∈ 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.

\[
P = e^{-\left(\frac{f(s') - f(s)}{T}\right)}
\]  
(3)

\[
\lim_{T \to \infty} - \left(\frac{f(s') - f(s)}{T}\right) = e^0 = 1
\]  
(4)

\[
\lim_{T \to 0} - \left(\frac{f(s') - f(s)}{T}\right) = e^{-\infty} = 0
\]  
(5)

Equation 3, 4 and 5 are the basic equations of SA. So firstly the possibility of bad solutions acceptance or (hill-climbing moves) is high for diversification. 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. The proper annealing process is related initial temperature, iteration for each temperature, temperature decrement coefficient and stopping criteria. All these criteria can be found at 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. More detailed information about GA can be found [7,8].

#### 5. Results

In this study it has been aimed to explain how constrained continuous optimization problems can be applied to unconstrained optimization solvers and to prove that metaheuristics can be used to solve continuous constraints problem solutions. As it has been seen in the Table 3, results are very near to real optimum value.

Since the problems are very basic, it has not been done solution time performance analysis in the study. But it can be advised genetic algorithms or simulated annealing for the solution of constrained continuous optimization problems, when the problem size is big or analytic solution method is difficult.
Figure 2.1. SA Flow Chart

Figure 2.2. GA Algorithm
Table 3. Problems solutions with analytic, SA and GA

<table>
<thead>
<tr>
<th>Problem No</th>
<th>Number of Variables</th>
<th>Real Optimum Function value and variables</th>
<th>GA</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$F_{opt}$ 82.5 X1 17.5 X2 15 X3</td>
<td>82.4621 X1 7.4989 X2 14.9975 -</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>668.333 X1 48.333 X2 0 X3</td>
<td>648.812 X1 48.3146 X2 0.04885 53.2486</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-3456 X1 24 X2 12 X3</td>
<td>-3434.1 X1 25.2076 X2 11.8989 12.0403</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-2.2137 X1 0.8165 X2 0.375 X3</td>
<td>-2.21366 X1 0.81680 X2 0.3749 -</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>19.8100 X1 -1.9888 X2 3.9818 X3</td>
<td>19.9357 X1 -1.9937 X2 3.9932 -</td>
<td></td>
</tr>
</tbody>
</table>

References


Electronics Engineering Turkey 2007;198.