Biased Random-key Genetic Algorithm for Linearly-constrained Global Optimization

Ricardo M. A. Silva
Centro de Informática, Universidade Federal de Pernambuco, Recife, PE, Brazil. email:rmas@cin.ufpe.br
Mauricio G. C. Resende
Algorithms and Optimization Research Dep., AT&T Labs Research, NJ, USA. email:mgcr@research.att.com
Panos M. Pardalos
Department of Industrial and Systems Engineering, University of Florida, FL, USA. email:pardalos@ufl.edu
João L. Facó
Instituto de Matemática, Universidade Federal do Rio de Janeiro, RJ, Brazil. email:jldfaco@ufrj.br

ABSTRACT

In this paper, we propose a biased random key genetic algorithm for finding approximate solutions for bound-constrained continuous global optimization problems subject to linear constraints. Experimental results illustrate its effectiveness on the `g01` and `g14` problems from CEC2006 benchmark [5].

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Global Optimization; I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic

Keywords

Linear constraints, global optimization, continuous optimization, heuristic, biased random key genetic algorithm.

1. INTRODUCTION

Continuous global minimization optimization seeks a solution $x^* \in S \subseteq R^n$ such that $f(x^*) \leq f(x)$, $\forall\, x \in S$, where S is some region of R^n and the objective function f is defined by $f : S \to R$. In this paper, we consider the domain S as the intersection between a set of linear constraints and a hyper-rectangle $\Omega = \{x = (x_1, \ldots, x_n) \in R^n : \ell \leq x \leq u\}$, where $\ell \in R^n$ and $u \in R^n$ such that $u_i \geq l_i$, for $i = 1, \ldots, n$, in order to present a BRKGA heuristic for bound-constrained continuous global optimization problems subject to linear constraints: min $f(x)$, subject to $Ax \leq b$ where $\Omega = \{x = (x_1, \ldots, x_n) \in R^n : \ell \leq x \leq u\}$, $A \in R^{m \times n}$ is the matrix whose rows are the vectors: $a_1 = (a_{1,1}, \ldots, a_{1,n}), \ldots, a_m = (a_{m,1}, \ldots, a_{m,n})$, and $b = (b_1, \ldots, b_m) \in R^m$.

Given that the constraints can be written as equalities with the introduction of the m slack variables $x_{n+1}, \ldots, x_{n+m}$: $(\sum_{j=1}^{n} a_{ij}x_j) + x_{n+i} = b_i, i = 1, \ldots, m$, with $x_k \geq 0, k = n+1, \ldots, n+m$, the original problem can be reduced to the following global optimization problem: min $F(x_1, \ldots, x_{n+m}) = [f(x_1, \ldots, x_n) - f^*]^2 + \sum_{i=1}^{m}[(\sum_{j=1}^{n} a_{ij}x_j) + x_{n+i} - b_i]^2$, subject to: $l_i \leq x_i \leq u_i, i = 1, \ldots, n$, and $x_k \geq 0, k = n+1, \ldots, n+m$, where f^* is a known optimum value of problem, or the best known value in the literature.

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

2. BIASED RANDOM-KEY GENETIC ALGS.

A BRKGA [2] evolves a population of random-key vectors over a number of iterations, called *generations*. The initial population is made up of p vectors of random-keys. Each component of the solution vector is generated independently at random in the real interval $[0, 1]$. After the fitness of each individual is computed by the decoder in generation k, the population is partitioned into two groups of individuals: a small group of p_e *elite* individuals, i.e. those with the best fitness values, and the remaining set of $p - p_e$ *non-elite* individuals. To evolve the population, a new generation of individuals must be produced. All elite individual of the population of generation k are copied without modification to the population of generation $k+1$. BRKGAs implement mutation by introducing *mutants* into the population. A mutant is simply a vector of random keys generated in the same way that an element of the initial population is generated. At each generation, a small number (p_m) of mutants is introduced into the population. With the p_e elite individuals and the p_m mutants accounted for population $k+1$, $p - p_e - p_m$ additional individuals need to be produced to complete the p individuals that make up the new population. This is done by producing $p - p_e - p_m$ offspring through the process of mating or crossover. The mechanism of mating in BRKGAs is the *parameterized uniform crossover* [6].

To describe a BRKGA for linearly-constrained global optimization problem, one needs to show how solutions are encoded and how these vectors are decoded to feasible solutions. A solution is encoded as a vector $\chi = (\chi_1, ..., \chi_n)$ of size n, where χ_i is a random number in the interval $[0, 1]$, for $i = 1, \ldots, n$. The i-th component of χ corresponds to the i-th dimension of hyper-rectangle Ω. A decoder takes as input the vector of random keys χ and returns a solution $x \in \Omega$ with $x_i = l_i + \chi_i \cdot (u_i - l_i)$, for $i = 1, \ldots, n$. After obtaining the solution $x \in \Omega$, we proceed by trying to improve it using the local search described in the next subsection. The solutions produced by the local search usually disagree with the genes initially supplied in the vector of random keys to the decoder. In these cases, in order to reflect the changes made by the local search phase of the decoder, the heuristic replaces the initial chromosome with the returned by the local search procedure, where $\chi_i = (x_i - l_i)/(u_i - l_i)$, for $i = 1, \ldots, n$.

2.1 Local improvement procedure

Let $\bar{x} \in \mathbb{R}^n$ be the current solution and h be the current grid discretization parameter. Define $S_h(\bar{x}) = \{x \in \Omega \mid \ell \leq x \leq u,\ x = \bar{x} + \tau \cdot h,\ \tau \in Z^n\}$ to be the set of points in Ω that are integer steps (of size h) away from $\bar{x}$. Let $B_h(\bar{x}) = \{x \in \Omega \mid x = \bar{x} + h \cdot (x' - \bar{x})/\|x' - \bar{x}\|,\ x' \in S_h(\bar{x}) \setminus \{\bar{x}\}\}$ be the projection of the points in $S_h(\bar{x}) \setminus \{\bar{x}\}$ onto the hyper-sphere centered at $\bar{x}$ of radius h. The *h-neighborhood* of the point $\bar{x}$ is defined as the set of points in $B_h(\bar{x})$. The procedure

```
procedure LocalImprovement(x, f(·), h_s, h_e, ℓ, u, MaxPointsToExamine)
1     x* ← x;
2     f* ← f(x);
3     h ← h_s;
4     Impr ← false;
5     while h ≥ h_e do
6           NumPointsExamined ← 0;
7           while NumPointsExamined ≤ MaxPointsToExamine do
8                 x ← RandomlySelectElement(B_h(x*));
9                 if ℓ ≤ x ≤ u and f(x) < f* then
10                      x* ← x;
11                      f* ← f(x);
12                      NumPointsExamined ← 0;
13                      Impr ← true;
14                end if
15                NumPointsExamined ← NumPointsExamined + 1;
16          end while
17          if Impr = true then
18                return x*;
19          else
20                h ← h/2;
21          end if
22    end while
23    return x*;
end LocalImprovement;
```

Figure 1: Pseudo-code for local improvement phase.

takes as input a starting solution $x \in \Omega \subseteq \mathbb{R}^n$, the objective function $f(\cdot)$, lower and upper bound vectors ℓ and u, as well as the parameters h_s and h_e, the starting and ending grid discretization densities, respectively. The maximum number of points `MaxPointsToExamine` $\leq \prod_{i=1}^{n} \lceil (u_i - \ell_i)/h \rceil$ in $B_h(x^*)$ that are to be examined is also taken as an input parameter. If all of these points are examined and no improving point is found, the current solution x^* is considered an *h-local minimum*.

The current best local improvement solution x^* is initialized to x in line 1. In line 2, the objective function value f^* of the best solution found is initialized to $f(x)$. Next, the parameter h, that controls the discretization density of the search space, is initialized to h_s in line 3, and in line 4 the variable `Impr` is set to **false**. Starting at the point x^*, in the loop in lines 7–16, the algorithm randomly selects points in $B_h(x^*)$ (line 8), one at a time. In line 9, if the current point x selected from $B_h(x^*)$ is feasible and is better than x^*, then x^* is set to x (line 10), f^* is set to $f(x)$ (line 11), `NumPointsExamined` is set to zero (line 12), `Impr` is set to **true** (line 13), and the loop in lines 7–16 restarts with x^* as the starting solution. In line 17, if the variable `Impr` is still set to false, then in line 20 the grid density is increased by halving h, and the loop in lines 7–16 is re-initialized if $h \geq h_e$. Local improvement is terminated if an *h-local minimum* solution x^* is found. At that point, x^* is returned from the local improvement procedure in line 18 or 23.

3. EXPERIMENTAL RESULTS

We made use of the unique linearly-constrained global optimization problems available in CEC2006 benchmark [5]: the test instances `g01` [3] and `g14` [4], with optimal values $f(x^*)$ equals to -15 and -47.76488 respectively.

In both problems, we ran BRKGA 200 times (a different starting random number seed for each run from 270001 to 270200) with $p = 100$, $p_e = 0.2p$, $p_m = 0.1p$, $\rho_e = 0.7$, $h_s = 0.05$, $h_e = 0.00001$, $rho_{lo} = 0.15$, `MaxPointsToExamine` = 1000, and $\epsilon = 0.00001$. At any time during a run, we define the optimality gap by $GAP = |F(x_1, \ldots, x_{n+m}) - F(z^*)|$, where $(x_1, \ldots, x_{n+m})$ is the current best solution found by the heuristic and $F(z^*) = 0$. We then say that the heuristic has solved the problem if $GAP \leq \epsilon$ with $\epsilon = 0.00001$.

We record the time taken to find the optimal (or best know) solution for each problem, in order to know its runtime distribution (or time-to-target plots [1]). While about 95% of the runs terminated in less than 45 seconds for `g01` problem, in `g14` instance 95% of the runs terminated in less than 653 seconds. In each problem, the BRKGA heuristic was able to find the optimal (or best known) solution in all 200 running, taking an average of 21.615 and 572.938 seconds to `g01` and `g14` problems, respectively. The minimum, mean, and maximum times, as well as the standard deviation, first, second (median), and third quartiles of the running times spent to find the solutions are (2.760, 13.238, 18.440, 21.615, 25.363, 253.620, 20.324) and (421.650, 539.688, 576.635, 572.938, 602.338, 681.390, 47.864) for `g01` and `g14` problems, respectively. In the worst (maximum) case, BRKGA found the target solutions for `g01` and `g14` problems in less than 254 and 682 seconds, respectively.

4. CONCLUDING REMARKS

In this paper, we present the BRKGA heuristic for finding approximate solutions for continuous global optimization problems subject to box and linear constraints. We illustrate the approach using the challenging `g01` and `g14` problems from CEC2006 benchmark [5]. The promising results shown here illustrate the potential of BRKGA for linearly-constrained global optimization problems.

5. ACKNOWLEDGMENTS

The research of R.M.A Silva was partially supported by the CNPq, FAPEMIG, CAPES, PROPESQ and FACEPE.

6. REFERENCES

[1] R. Aiex, M. Resende, and C. Ribeiro. Prob. distribution of solution time in GRASP: An experimental investigation. *J. of Heuristics*, 8:343–373, 2002.

[2] M. Ericsson, M. Resende, and P. Pardalos. A genetic algorithm for the weight setting prob. in OSPF routing. *J. of Combinatorial Optimization*, 6:299–333, 2002.

[3] C. Floudas and P. Pardalos. *Collection of test probs. for constrained global optimization algs.* Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[4] D. Himmelblau. *Applied nonlinear programming.* McGraw-Hill, 1972.

[5] J. J. Liang, T. P. Runarsson, E. M. Montes, M. Clerc, P. N. Suganthan, C. A. Coello, and D. K. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, 2006.

[6] W. Spears and K. DeJong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.