Random-key genetic algorithms

José Fernando Gonçalves

LIAAD, INESC TEC, Faculdade de Economia do Porto, Universidade do Porto

Porto, Portugal

Mauricio G. C. Resende

Network Evolution Research Department, AT&T Labs Research

Middletown, New Jersey, U.S.A.

August 18, 2014

Abstract

A random-key genetic algorithm is an evolutionary metaheuristic for discrete and global optimization. Each solution is encoded as a vector of n random keys, where a random key is a real number, randomly generated, in the continuous interval $[0, 1)$. A decoder maps each vector of random keys to a solution of the optimization problem being solved and computes its cost. The algorithm starts with a population of p vectors of random keys. At each iteration, the vectors are partitioned into two sets, a smaller set of high-valued elite solutions, and the remaining non-elite solutions. All elite elements are copied, without change, to the next population. A small number of random-key vectors (the mutants) is added to the population of the next iteration. The remaining elements of the population of the next iteration are generated by combining, with the parametrized uniform crossover of Spears and DeJong [58], pairs of solutions. This chapter reviews random-key genetic algorithms and describes an effective variant called biased random-key genetic algorithms.

Keywords – Genetic algorithm, Random keys, Optimization.

1 Genetic algorithm with random keys

Bean [6] described a new class of genetic algorithms for combinatorial optimization problems whose solutions can be represented by a permutation vector. These algorithms, called *random-key genetic algorithms* (RKGA), represent a solution of the optimization problem as a vector of random keys. A *random key* is a real number, generated at random in the continuous interval $[0, 1)$.

A *decoder* is a procedure that maps a vector of random keys into a solution of the optimization problem and computes the cost of this solution. The decoder proposed by Bean [6] simply orders the elements of the vector of random keys, thus producing a permutation corresponding to the indices of the sorted elements.

A RKGA evolves a population, or set, of p vectors of random keys applying the Darwinian principle of survival of the fittest, where the fittest individuals (or solutions) of a population are more likely to find a mate and pass on their genetic material to future generations. The algorithm starts with an initial population of p vectors of n random keys and produces a series of populations. In the k-th generation, the p vectors of the population are partitioned into a small set of $p_e < p/2$ vectors corresponding to the best solutions (this set is called the *elite* set) and another set with the remainder of the population (called the *non-elite* set). All elite vectors are copied, unchanged, to the population of the $k + 1$-st generation. This elitism characterizes the Darwinian principle in an RKGA. Next, p_m vectors of random keys are introduced into the population of the $k + 1$-st generation. These vectors, called *mutants* or *immigrants*, and play the same role as the mutation operators of classical genetic algorithms, i.e. they help avoid convergence of the population to a non-global local optimum. To complete the p elements of the population of the $k+1$-st generation, $p - p_e - p_m$ vectors are generated, combining pairs of solutions from the population of the k-th generation, with a parametrized uniform crossover [58]. Let a and b be the vectors chosen for mating and let c be the offspring produced. In the crossover of Spears and DeJong [58], $c[i]$, the i-th component of the offspring vector, receives the i-th key of one of its parents. It receives the key $a[i]$ with probability ρ_a and $b[i]$ with probability $\rho_b = 1 - \rho_a$.

2 Biased random-key genetic algorithms

As seen in Section 1 of this chapter, Bean's algorithm limits itself to elitism to simulate Darwinism. A *biased random-key genetic algorithm* (or BRKGA [24]), on the other hand, not only uses elitism to simulate survival of the fitness, but also makes use of mating. A BRKGA differs from Bean's algorithm in the way parents are selected for crossover and how crossover is applied.

Both algorithms choose parents at random and with replacement. This way a parent can have more than one offpring per generation. While in Bean's algorithm both parents are chosen from the entire population, in a BRKGA one parent is always chosen from the elite set while the other is chosen from the non-elite set (or, in some cases, from the entire population). Since $p_e < p/2$, an elite vector in a BRKGA has a probability of $1/p_e$ of being selected for each crossover. This is greater than $1/(p - p_e)$, the probability that a non-elite vector has of being selected. For the same reason, the probability that a specific elite vector is chosen in a BRKGA is greater than $1/p$, the probability that a given elite vector is chosen in Bean's algorithm.

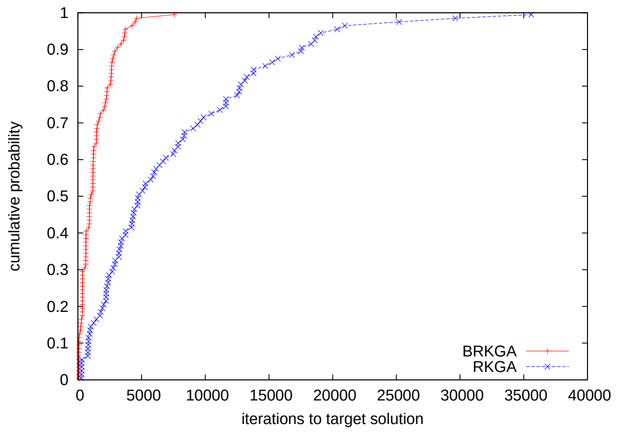

Figure 1: Iteration count distributions to a given target solution value for a BRKGA and Bean's algorithm.

Both algorithms combine parents a and b using parametrized uniform crossover [58] to produce the offspring c. While in Bean's algorithm each parent can be parent a or b, in a BRKGA a is always the elite parent and b is the non-elite parent. Since $\rho_a > 1/2$, in a BRKGA the offspring c has greater probability of inheriting the keys of the elite parent, while in Bean's algorithm this is not necessarily true.

This small difference between the two algorithms almost always results in BRKGA outperforming Bean's algorithm [33]. Figure 1 compares iteration count distributions to a given target value for a BRKGA and an implementation of Bean's for a covering by pairs problem [7]. The figure clearly shows the dominance of the biased variant of the RKGA over the unbiased variant on this instance and for this target value. Though there has been at least one instance where the unbiased variant was slightly superior to the biased variant, the dominance of the biased variant over the unbiased variant is well established [33].

3 A model for the implementation of a BRKGA

Algorithm 1 shows a pseudo-code of a BRKGA for the minimization of $f(x)$, where $x \in X$ and X is a discrete set of solutions and $f : X \to \mathbb{R}$. This implementation is a multi-start variant of a BRKGA where several populations are evolved in sequence and a best solution among all in the population is returned as the output of the algorithm. After describing the pseudo-code, we will justify its multi-start nature.

In line 2, the value f^* of the best solution found is initialized to a large value, i.e. not smaller than $f(x^0)$, where $x^0 \in X$ is some feasible solution to the problem. Evolution of each population is done in lines 3 to 28. The algorithm halts when some stopping criterion in line 3 is satisfied. This criterion can be, for example, number of evolved populations, total time, or quality of the best solution found.

In line 4, the population being evolved is initialized with $p = |\mathcal{P}|$ vectors of random keys. Evolution of population $\mathcal{P}$ takes place in lines 5 to 27. This evolution ends when a restart criterion is satisfied in line 5. This criterion can be, for example, a maximum number of generations without improvement in the value of the best solution in $\mathcal{P}$. At each generation, or iteration, the following operations are carried out: In line 6 all new solutions (offspring and mutants) are decoded and their costs evaluated. Note that each decoding and evaluation in this step can be computed simultaneously, i.e. in parallel. In line 7, population $\mathcal{P}$ is partitioned into two subpopulations $\mathcal{P}_e$ (elite) and $\mathcal{P}_{\bar{e}}$ (non-elite), where $\mathcal{P}_e$ is such that $|\mathcal{P}_e| < |\mathcal{P}|/2$ and contains $|\mathcal{P}_e|$ of the best solutions in $\mathcal{P}$ and $\mathcal{P}_{\bar{e}}$ consists of the remaining solutions in $\mathcal{P}$, that is

```
BRKGA($|\mathcal{P}|, |\mathcal{P}_e|, |\mathcal{P}_m|, n, \rho_a$)
Initialize value of the best solution found: $f^* \leftarrow \infty$;
while stopping criterion not satisfied do
    Generate a population $\mathcal{P}$ with $n$ vectors of random keys;
    while restart criterion not satisfied do
        Evaluate the cost of each new solution in $\mathcal{P}$;
        Partition $\mathcal{P}$ into two sets: $\mathcal{P}_e$ and $\mathcal{P}_{\bar{e}}$;
        Initialize population of next generation: $\mathcal{P}^+ \leftarrow \mathcal{P}_e$;
        Generate set $\mathcal{P}_m$ of mutants, each mutant with $n$
        random keys;
        Add $\mathcal{P}_m$ to population of next generation:
        $\mathcal{P}^+ \leftarrow \mathcal{P}^+ \cup \mathcal{P}_m$;
        foreach $i \leftarrow 1$ to $|\mathcal{P}| - |\mathcal{P}_e| - |\mathcal{P}_m|$ do
            Select parent $a$ at random from $\mathcal{P}_e$;
            Select parent $b$ at random from $\mathcal{P}_{\bar{e}}$;
            foreach $j \leftarrow 1$ to $n$ do
                Throw a biased coin with probability
                $\rho_a > 0.5$ of resulting heads;
                if heads then $c[j] \leftarrow a[j]$ ;
                else $c[j] \leftarrow b[j]$;
            end
            Add offspring $c$ to population of next
            generation: $\mathcal{P}^+ \leftarrow \mathcal{P}^+ \cup \{c\}$;
        end
        Update population: $\mathcal{P} \leftarrow \mathcal{P}^+$;
        Find best solution $\chi^+$ in $\mathcal{P}$:
        $\chi^+ \leftarrow \mathbf{argmin}\{f(\chi) \mid \chi \in \mathcal{P}\}$;
        if $f(\chi^+) < f^*$ then
            $\chi^* \leftarrow \chi^+$;
            $f^* \leftarrow f(\chi^+)$;
        end
    end
end
return $\chi^*$
```

Algorithm 1: Model for biased random-key genetic algorithm with restart.

$\mathcal{P}_{\bar{e}} = \mathcal{P} \setminus \mathcal{P}_e$. $\mathcal{P}^+$ is the population of the next generation. It is initialized in line 8 with the elite solutions of the current generation. In line 9, the mutant subpopulation $\mathcal{P}_m$ is generated. Each mutant is a vector of n random keys. The number of generated mutants in general is such that $|\mathcal{P}_m| < |\mathcal{P}|/2$. This subpopulation is added to population $\mathcal{P}^+$ of the next generation in line 10.

With $|\mathcal{P}_e| + |\mathcal{P}_m|$ vectors inserted in population $\mathcal{P}^+$, it is necessary to generate $|\mathcal{P}| - |\mathcal{P}_2| - |\mathcal{P}_m|$ new offspring to complete the $|\mathcal{P}|$ vectors that form population $\mathcal{P}^+$. This is done in lines 11 to 20. In lines 12 and 13 parents a and b are chosen, respectively, at random from subpopulations $\mathcal{P}_e$ and $\mathcal{P}_{\bar{e}}$. The generation of offspring c from parents a and b takes place in lines 14 to 18. A biased coin (with probability $\rho_a > 1/2$ of flipping to heads) is thrown n times. If the i-th toss is a heads, the offspring inherits the i-th key of parent a. Otherwise, it inherits the i-th key of parent b. After the offspring is generated, c is added to population $\mathcal{P}^+$ in line 19.

The generation of $\mathcal{P}^+$ ends when it consists of $|\mathcal{P}|$ elements. In line 21, $\mathcal{P}^+$ is copied to $\mathcal{P}$ to start a new generation. The best solution in the current population in evolution is computed in line 22 and if its value is better than all solutions examined so far, the solution and its cost are saved in lines 24 and 25 as χ^* and f^*, respectively. χ^*, the best solution found over all populations is returned by the algorithm in line 29.

4 Restarting a random-key genetic algorithm

As with most stochastic search methods, the continuous random variable *time to target solution* of a RKGA has an empirical distribution that approximates a shifted exponential distribution. The discrete random variable *iterations to target solution*, on the other hand, has an empirical shifted

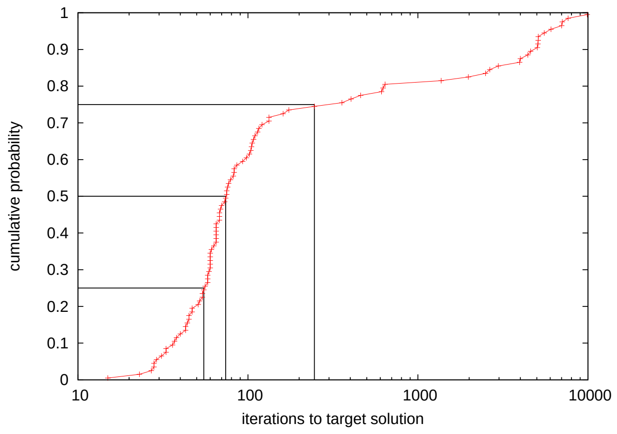

Figure 2: Iteration count distribution to target solution of a BRKGA without restart.

geometric distribution.

Consider, in Figure 2, the empirical distribution of number of iterations of a BRKGA to find an optimal solution of instance of Steiner triple covering problem `stn243` [50].

The *iterations-to-target-solution plot* [1] is generated by running the BRKGA 100 times, each time using a different seed for the random number generator and recording the number of iterations that the algorithm took to find a solution as least as good as the target (in this case an optimal solution). The figure shows that 25% of the runs needed no more than 55 iterations to find an optimal solution, 50% took at most 74 iterations and 75% at most 245. However, 10% of the runs required more than 4597 iterations, 5% more than 5532 iterations, 2% more than 7061 and the longest run took 9903 iterations. This is the typical behavior of a random variable com a shifted geometric distribution.

Let I be the random variable number of iterations to a given target solution. For instance `stn243`, a visual examination of Figure 2 suggests that $\Pr(I \geq 246) \approx 1/4$. Restarting the algorithm after 246 iterations and assuming independence of the runs, $\Pr(I \geq 492 \mid I \geq 246) \approx 1/4$. Therefore, $\Pr(I \geq 492) = \Pr(I \geq 246) \times \Pr(I \geq 492 \mid I \geq 246) \approx 1/4^2$. One can easily show, by induction, that the probability that the algorithm will take fewer than k cycles of 246 iterations is approximately $1/4^k$. For example, the probability that the algorithm with restart will take more than 1230 iterations (five cycles of 246 iterations between restarts) is approximately $1/4^5 = 1/1024 \approx 0.1\%$. This probability is considerably smaller than the approximately 20% probability that the algorithm without restart will take more than 1230 iterations.

The above analysis uses a restart strategy that differs slightly from the one proposed here for random-key genetic algorithms. In the proposed strategy, similar to the restart strategy for GRASP with path-relinking proposed by Resende and Ribeiro [49], instead of restarting each k iterations, it restarts after k_r iterations without improvement of the value of the best solution found since the previous restart.

Figure 3 compares a BRKGA without restart with one which restarts every 246 iterations without improvement of the value of the best solution found on Steiner triple covering instance `stn243`. The figure clearly shows that both the average number of iterations to an optimum as well as the corresponding standard deviation are smaller in the variant with restart than in the one without restart.

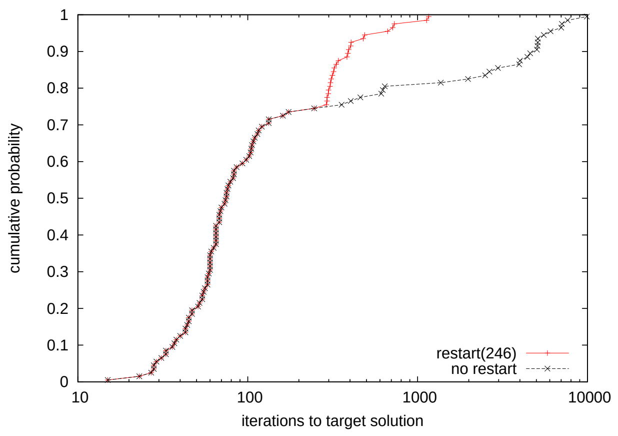

Figure 3: Iteration count distribution to a target (optimal) solution of variants of BRKGA with and without restart on Steiner triple covering instance `stn243`.

5 RKGA with multiple populations

The description of random-key genetic algorithms so far involved a single population. However, it is possible to implement a RKGA with more than one population [25].

Suppose that the RKGA has π populations, $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\pi$, each with p vectors of random keys. In this case, the π populations are initially populated, each independently of the others, with p vectors of random-keys in line 4 of the pseudo-code of Algorithm 1 and the loop in lines 6 to 26 is applied to each of these π populations. The populations exchange information every k_p iterations of the loop in lines 5 to 27 of Algorithm 1. In this exchange, the k_m best solutions from each population replace the $(\pi - 1)k_m$ worst solutions of each population.

6 Specifying a RKGA

The specification of a RKGA requires defining how a solution is represented, or encoded, how decoding is done, and what are the parameters of the algorithm.

Since each solution is represented as a vector of n random keys, it is necessary only to specify a value for n.

The decoder is a deterministic algorithm that takes as input a vector of n random keys and produces as output a solution of the optimization problem as well as its corresponding cost. A decoder is, in general, a heuristic. If it makes use of local search, then it is recommended but not strictly necessary that a vector adjustment procedure be specified such that when the decoder is applied to a vector of random keys corresponding to a local optimum the decoder will produce the local optimum without applying the local search phase of the decoder. See Resende et al. [50] for a simple example of vector adjustment.

Several parameters need to be specified. Table 6 lists these parameters and offers value ranges which in practice have proven to be satisfactory [24].

7 API for BRKGA

To simplify the implementation of BRKGAs, Toso and Resende [61] proposed an *Application Programming Interface (API)*, or C++ library, for BRKGA. The API is efficient and easy to use. The library is portable and automatically deals with several aspects of the BRKGA, such as management of the population and of the evolutionary dynamics. The API is implemented in C++ and uses an object oriented architecture. In systems

Table 1: Parameters and recommended values.

Parameter	**Recommended value**
p: size of population	$p = \max\{3, \lfloor \kappa_p \times n \rfloor\}$, where $\kappa_p > 0$
p_e: size of elite partition of population	$p_e = \max\{1, \lfloor \kappa_e \times p \rfloor\}$, where $\kappa_e \in [0.10, 0.25]$
p_m: size of mutant partition of population	$p_m = \max\{1, \lfloor \kappa_m \times p \rfloor\}$, where $\kappa_m \in [0.05, 0.20]$
ρ_a: probability of inheriting key from elite parent	$\rho_a > 1/2$,
k_r: iterations without improvement for restart	$k_r = \text{argmin}\{\Pr(k \text{ iterations to target solution}) \geq 0.75\}$
π: number of parallel populations	$\pi \in \{1, \ldots, 5\}$
k_p: frequency for population interchange	$k_p \in \{50, \ldots, 100\}$
k_m: number of exchanged solutions	$k_m \in \{1, 2, 3\}$
stopping criterion (examples)	running time, maximum number of iterations, maximum number of restarts, finding a solutions as good as the target.

with available *OpenMP* [44], the API enables parallel decoding of random key vectors. The user only needs to implement the decoder and specify the stopping criteria, restart and population exchange mechanisms, as well as the parameters of the algorithm.

The API is open source and can be downloaded from `http://github.com/rfrancotoso/brkgaAPI`.

8 Final remarks

This chapter reviewed random-key genetic algorithms, covering both their unbiased and the biased variants. After introducing the algorithm of Bean [6], on which the BRKGA is based, the chapter points to two small differences between the two variants that lead to improved performance of the BRKGA with respect to Bean's original random-key genetic algorithm. A model for the implementation of a BRKGA is described and issues such as restart and use of multiple populations are discussed. The chapter concludes by illustrating how a BRKGA is specified and presents an C++ API for BRKGA that allows for easy implementation of the algorithm.

The BRKGA metaheuristic has been applied to many optimization problems, such as:

- *Telecommunications:* Ericsson et al. [15], Buriol et al. [8], Noronha et al. [43], Noronha et al. [43], Reis et al. [47], Ruiz et al. [53], Pedrola et al. [46], Goulart et al. [34], Resende [48], Morán-Mirabal et al. [40], Pedrola et al. [45], Duarte et al. [14], and Andrade et al. [2].
- *Transportation:* Buriol et al. [10], Grasas et al. [35], Stefanello et al. [59], and Lalla-Ruiz et al. [36].

- *Scheduling:* Gonçalves et al. [29], Valente et al. [63], Valente and Gonçalves [62], Gonçalves et al. [30], Mendes et al. [38], Gonçalves et al. [31], Tangpattanakul et al. [60], Gonçalves and Resende [28], and Marques et al. [37].
- *Packing:* Gonçalves [20], and Gonçalves and Resende [25, 26, 27].
- *Clustering:* Festa [16] and Andrade et al. [4].
- *Covering:* Breslau et al. [7], and Resende et al. [50].
- *Network optimization:* Andrade et al. [5], Buriol et al. [9], Fontes and Gonçalves [19], Coco et al. [12], Fontes and Gonçalves [18], Ruiz et al. [52], Andrade et al. [2], and Coco et al. [13].
- *Power Systems:* Roque et al. [51].
- *Industrial Engineering:* Gonçalves and Beirão [22], Gonçalves and Almeida [21], Gonçalves and Resende [23], Moreira et al. [42], Morán-Mirabal et al. [41], Gonçalves et al. [32], and Chan et al. [11].
- *Automatic tuning of parameters in heuristics:* Festa et al. [17], and Morán-Mirabal et al. [39].
- *Combinatorial auctions:* Andrade et al. [3].
- *Global continuous optimization:* Silva et al. [54], Silva et al. [56, 55], and Silva et al. [57]

Acknowledgement

The first author was partially supported by funds granted by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT – Foundation for Science and Technology, project PTDC/ EGE-GES/ 117692/ 2010.

References

[1] R. M. Aiex, M. G. C. Resende, and C. C. Ribeiro. TTTPLOTS: A perl program to create time-to-target plots. *Optimization Letters*, 1: 355–366, 2007.

[2] C. E. Andrade, F. K. Miyazawa, and M. G .C. Resende. Evolutionary algorithm for the k-interconnected multi-depot multi-traveling salesmen problem. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2013.

[3] C. E. Andrade, F. K. Miyazawa, M. G. C. Resende, and R.F. Toso. Biased random-key genetic algorithms for the winner determination problem in combinatorial auctions. Technical report, AT&T Labs Research, Florham Park, New Jersey, 2013.

[4] C.E. Andrade, M.G.C. Resende, H.J. Karloff, and F.K. Miyazawa. Evolutionary algorithms for overlapping correlation clustering. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'14)*, pages 405–412, Vancouver, Canada, July 2014.

[5] D. V. Andrade, L. S. Buriol, M. G. C. Resende, and M. Thorup. Survivable composite-link IP network design with OSPF routing. In *Proceedings of The Eighth INFORMS Telecommunications Conference*, 2006.

[6] J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA J. on Computing*, 6:154–160, 1994.

[7] L. Breslau, I. Diakonikolas, N. Duffield, Y. Gu, M. Hajiaghayi, D. S. Johnson, H. Karloff, M. G. C. Resende, and S. Sen. Disjoint-path facility location: Theory and practice. In *Proceedings of the Thirteenth Workshop of Algorithm Engineering and Experiments (ALENEX11)*, pages 68–74, 2011.

[8] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 46:36–56, 2005.

[9] L. S. Buriol, M. G. C. Resende, and M. Thorup. Survivable IP network design with OSPF routing. *Networks*, 49:51–64, 2007.

[10] L. S. Buriol, M. J. Hirsch, T. Querido, P. M. Pardalos, M. G. C. Resende, and M. Ritt. A biased random-key genetic algorithm for road congestion minimization. *Optimization Letters*, 4:619–633, 2010.

[11] F. T. S. Chan, R. K. Tibrewal, A. Prakash, and M. K. Tiwari. A biased random key genetic algorithm approach for inventory-based multi-item lot-sizing problem. *J. of Engineering Manufacture*, 2014. Published online March 20, 2014.

[12] A. A. Coco, T. F. Noronha, and A. C. Santos. A biased random-key genetic algorithm for the robust shortest path problem. In *Proceedings of Global Optimization Workshop (GO2012)*, pages 53–56, 2012.

[13] A. A. Coco, J. C. A. Abreu Jr., T. F. Noronha, and A. C. Santos. An integer linear programming formulation and heuristics for the minmax relative regret robust shortest path problem. *J. of Global Optimization*, 2014. Published online April 22, 2014.

[14] A. Duarte, R. Martí, M. G. C. Resende, and R.M.A. Silva. Improved heuristics for the regenerator location problem. *International Transactions in Operational Research*, 21:541–558, 2014.

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

[16] P. Festa. A biased random-key genetic algorithm for data clustering. *Mathematical Biosciences*, 245:76–85, 2013.

[17] P. Festa, J. F. Gonçalves, M. G. C. Resende, and R. M. A. Silva. Automatic tuning of GRASP with path-relinking heuristics with a biased random-key genetic algorithm. In P. Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 338–349. Springer, 2010.

[18] D. B. M. M. Fontes and J. F. Gonçalves. A multi-population hybrid biased random key genetic algorithm for hop-constrained trees in non-linear cost flow networks. *Optimization Letters*, 2012. Publicado online 13 de junho.

[19] D. B. M. M. Fontes and J. F. Gonçalves. Heuristic solutions for general concave minimum cost network flow problems. *Networks*, 50:67–76, 2007.

[20] J. F. Gonçalves. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *European J. of Operational Research*, 183:1212–1229, 2007.

[21] J. F. Gonçalves and J. Almeida. A hybrid genetic algorithm for assembly line balancing. *J. of Heuristics*, 8:629–642, 2002.

[22] J. F. Gonçalves and N. C. Beirão. Um algoritmo genético baseado em chaves aleatórias para sequenciamento de operações. *Revista Associação Portuguesa de Desenvolvimento e Investigação Operacional*, 19: 123–137, 1999.

[23] J. F. Gonçalves and M. G. C. Resende. An evolutionary algorithm for manufacturing cell formation. *Computers and Industrial Engineering*, 47:247–273, 2004.

[24] J. F. Gonçalves and M. G. C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *J. of Heuristics*, 17:487–525, 2011.

[25] J. F. Gonçalves and M. G. C. Resende. A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. *J. of Combinatorial Optimization*, 22:180–201, 2011.

[26] J. F. Gonçalves and M. G. C. Resende. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers and Operations Research*, 29:179–190, 2012.

[27] J. F. Gonçalves and M. G. C. Resende. A biased random-key genetic algorithm for a 2D and 3D bin packing problem. *International J. of Production Economics*, 145:500–510, 2013.

[28] J. F. Gonçalves and M. G. C. Resende. An extended Akers graphical minimization method with a biased random-key genetic algorithm

for job-shop scheduling. *International Transactions in Operational Research*, 21:215–246, 2014.

[29] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European J. of Operational Research*, 167:77–95, 2005.

[30] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende. A genetic algorithm for the resource constrained multi-project scheduling problem. *European J. of Operational Research*, 189:1171–1190, 2008.

[31] J. F. Gonçalves, M. G. C. Resende, and J. J. M. Mendes. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *J. of Heuristics*, 17:467–486, 2011.

[32] J. F. Gonçalves, M. G. C. Resende, and M. D. Costa. A biased random-key genetic algorithm for the minimization of open stacks problem. *International Transactions in Operational Research*, 2014. Published online July 2, 2014.

[33] J.F. Gonçalves, M.G.C. Resende, and R.F. Toso. An experimental comparison of biased and unbiased random-key genetic algorithms. *Pesquisa Operacional*, 34:143–164, 2014.

[34] N. Goulart, S. R. de Souza, L. G. S. Dias, and T. F. Noronha. Biased random-key genetic algorithm for fiber installation in optical network optimization. In *IEEE Congress on Evolutionary Computation (CEC 2011)*, pages 2267–2271. IEEE, 2011.

[35] A. Grasas, H. R. Lourenço, L. S. Pessoa, M. G. C. Resende, I. Caballé, and N. Barba. On the improvement of blood sample collection at clinical laboratories. *BMC Health Services Research*, 14, 2014. Article 12.

[36] E. Lalla-Ruiz, J. L. González-Velarde, B. Melián-Batista, and J. M. Moreno-Vega. Biased random key genetic algorithm for the tactical berth allocation problem. *Applied Soft Computing*, 22:60–76, 2014.

[37] I. Marques, M. E. Captivo, and M. Vaz Pato. Scheduling elective surgeries in a portuguese hospital using a genetic heuristic. *Operations Research for Health Care*, 3:59–72, 2014.

[38] J. J. M. Mendes, J. F. Gonçalves, and M. G. C. Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 36:92–109, 2009.

[39] L. F. Morán-Mirabal, J. L. González-Velarde, and M. G. C. Resende. Automatic tuning of GRASP with evolutionary path-relinking. In *Proceedings of Hybrid Metaheuristics 2013 (HM 2013)*, volume 7919 of *Lecture Notes in Computer Science*, pages 62–77. Springer, Ischia, Italy, 2013.

[40] L. F. Morán-Mirabal, J. L. González-Velarde, M. G. C. Resende, and R. M. A. Silva. Randomized heuristics for handover minimization in mobility networks. *J. of Heuristics*, 19:845–880, 2013.

[41] L. F. Morán-Mirabal, J. L. González-Velarde, and M. G. C. Resende. Randomized heuristics for the family traveling salesperson problem. *International Transactions in Operational Research*, 21:41–57, 2014.

[42] M. C. O. Moreira, M. Ritt, A. M Costa, and A. A. Chaves. Simple heuristics for the assembly line worker assignment and balancing problem. *J. of Heuristics*, 18:505–524, 2012.

[43] T. F. Noronha, M. G. C. Resende, and C. C. Ribeiro. A biased random-key genetic algorithm for routing and wavelength assignment. *J. of Global Optimization*, 50:503–518, 2011.

[44] OpenMP, 2013. `http://openmp.org/wp/`, last visted on May 11, 2013.

[45] O. Pedrola, D. Careglio, M. Klinkowski, L. Velasco, K. Bergman, and J. Solé-Pareta. Metaheuristic hybridizations for the regenerator placement and dimensioning problem in sub-wavelength switching optical networks. *European J. of Operational Research*, 224:614–624, 2013.

[46] O. Pedrola, M. Ruiz, L. Velasco, D. Careglio, O. González de Dios, and J. Comellas. A GRASP with path-relinking heuristic for the survivable IP/MPLS-over-WSON multi-layer network optimization problem. *Computers and Operations Research*, 40:3174–3187, 2013.

[47] R. Reis, M. Ritt, L. S. Buriol, and M. G. C. Resende. A biased random-key genetic algorithm for OSPF and DEFT routing to minimize network congestion. *International Transactions in Operational Research*, 18: 401–423, 2011.

[48] M. G. C. Resende. Biased random-key genetic algorithms with applications in telecommunications. *TOP*, 20:120–153, 2012.

[49] M. G. C. Resende and C. C. Ribeiro. Restart strategies for GRASP with path-relinking heuristics. *Optimization Letters*, 5:467–478, 2011.

[50] M. G. C Resende, R. F. Toso, J. F. Gonçalves, and R. M. A Silva. A biased random-key genetic algorithm for the Steiner triple covering problem. *Optimization Letters*, 6:605–619, 2012.

[51] L. A. C. Roque, D. B. M. M. Fontes, and F. A. C. C. Fontes. A hybrid biased random key genetic algorithm approach for the unit commitment problem. *J. of Combinatorial Optimization*, 28:140–166, 2014.

[52] E. Ruiz, M. Albareda-Sambola, E. Fernández, and M.G.C. Resende. A biased random-key genetic algorithm for the capacitated minimum spanning tree problem. Technical report, AT&T Labs Research Technical, Florham Park, New Jersey, 2013.

[53] M. Ruiz, O. Pedrola, L. Velasco, D. Careglio, J. Fernández-Palacios, and G. Junyent. Survivable IP/MPLS-over-WSON multilayer network optimization. *J. of Optical Communications and Networking*, 3:629–640, 2011.

[54] R. M. A. Silva, M. G. C. Resende, P. M. Pardalos, and J. F. Gonçalves. Biased random-key genetic algorithm for bound-constrained global optimization. In *Proceedings of Global Optimization Workshop (GO2012)*, pages 133–136, 2012.

[55] R. M. A. Silva, M. G. C. Resende, and P. M. Pardalos. Finding multiple roots of box-constrained system of nonlinear equations with a biased random-key genetic algorithm. *J. of Global Optimization*, 2013. Published online September 13, 2013.

[56] R. M. A. Silva, M. G. C. Resende, and P.M. Pardalos. A Python/C++ library for bound-constrained global optimization using biased random-

key genetic algorithm. *J. of Combinatorial Optimization*, 2013. Published online October 5, 2013.

[57] R.M.A. Silva, M.G.C. Resende, P.M. Pardalos, and J.L.D. Facó. Biased random-key genetic algorithm for non-linearly constrained global optimization. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC)*, pages 2201–2206, Cancun, June 2013.

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

[59] F. Stefanello, L. S. Buriol, M. J. Hirsch, P. M. Pardalos, T. Querido, M.G.C. Resende, and M. Ritt. On the minimization of traffic congestion in road networks with tolls. Technical report, AT&T Labs Research, Florham Park, New Jersey, 2013.

[60] P. Tangpattanakul, N. Jozefowiez, and P. Lopez. Multi-objective optimization for selecting and scheduling observations by agile earth observing satellites. In *Parallel Problem Solving from Nature – PPSN XII*, volume 7492 of *Lecture Notes in Computer Science*, pages 112–121. Springer, 2012.

[61] R. F. Toso and M. G. C. Resende. A C++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 2014. Published online March 13, 2014.

[62] J. M. S. Valente and J. F. Gonçalves. A genetic algorithm approach for the single machine scheduling problem with linear earliness and

quadratic tardiness penalties. *Computers and Operations Research*, 35: 3696–3713, 2008.

[63] J. M. S. Valente, J. F. Gonçalves, and R. A. F. S. Alves. A hybrid genetic algorithm for the early/tardy scheduling problem. *Asia-Pacific J. of Operational Research*, 23:393–405, 2006.