AUTOMATIC TUNING OF GRASP WITH PATH-RELINKING HEURISTICS WITH A BIASED RANDOM-KEY GENETIC ALGORITHM

P. FESTA, J.F. GONÇALVES, M.G.C. RESENDE, AND R.M.A. SILVA

Abstract. GRASP with path-relinking (GRASP+PR) is a metaheuristic for finding optimal or near-optimal solutions of combinatorial optimization problems. This paper proposes a new automatic parameter tuning procedure for GRASP+PR heuristics based on a biased random-key genetic algorithm (BRKGA). Given a GRASP+PR heuristic with n input parameters, the tuning procedure makes use of a BRKGA in a first phase to explore the parameter space and set the parameters with which the GRASP+PR heuristic will run in a second phase. The procedure is illustrated with a GRASP+PR for the generalized quadratic assignment problem with $n = 30$ parameters. Computational results show that the resulting hybrid heuristic is robust.

1. Introduction

A commonly cited drawback of heuristics is the large number of parameters that need to be tuned for good performance. These parameters are not limited to those that are numerically valued but can also be logical parameters that determine, for example, which sub-modules are activated in the heuristic and which ones are not. Heuristic parameters can consequently run into the tens and even hundreds and tuning them can be a labor intensive activity. Furthermore, the performance of a heuristic depends on the instance being solved, so a tuned set of parameters obtained for one instance may not result in a good performing heuristic for another instance. When documenting a heuristic, a description of the tuning process is often left out and therefore it is often difficult to reproduce computational results. These are some of the factors that point to the need for an algorithmic approach to parameter tuning.

In this paper we propose an automatic tuning procedure for GRASP with path-relinking (GRASP+PR) heuristics. In the first phase of this two-phase solution strategy a biased random-key genetic algorithm searches the space of parameters for a set of values that results in a good performance of the heuristic. In the second phase, the GRASP+PR heuristic is run using the parameters found in the first phase. We illustrate this procedure on a GRASP+PR heuristic for the generalized quadratic assignment problem. This GRASP+PR has 30 tunable parameters. Computational results show that the two-phase approach results in a robust hybrid heuristic.

Date: February 16, 2010.

Key words and phrases. Automatic tuning of parameters, GRASP, path-relinking, algorithm engineering, metaheuristics.

AT&T Labs Research Technical Report.

The paper is organized as follows. In Section 2 we briefly describe the genetic algorithm. In Section 3 we summarize the solution strategy implemented in the GRASP+PR heuristic. The two-phase strategy is described in Section 4. Finally, computational experiments are reported in Section 5.

2. Biased random-key genetic algorithms

Genetic algorithms with random keys, or *random-key genetic algorithms* (RKGA), were first introduced by Bean [1] for solving combinatorial optimization problems involving sequencing. In a RKGA, chromosomes are represented as vectors of randomly generated real numbers in the interval $[0, 1]$. A deterministic algorithm, called a *decoder*, takes as input a solution vector and associates with it a solution of the combinatorial optimization problem for which an objective value or fitness can be computed.

A RKGA 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 $p_e = 0.3p$ *elite* individuals, i.e. those with the best fitness values, and the remaining set of $p-p_e = 0.7p$ *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 $k+1$. RKGAs 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 ($p_m = 0.2p$) of mutants is introduced into the population. With the p_e elite individuals and the p_m mutants accounted for in 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.

Bean [1] selects two parents at random from the entire population to implement mating in a RKGA. A *biased random-key genetic algorithm,* or BRKGA [2], differs from a RKGA in the way parents are selected for mating. In a BRKGA, each element is generated combining one element selected at random from the elite partition in the current population and one from the non-elite partition. Repetition in the selection of a mate is allowed and therefore an individual can produce more than one offspring in the same generation. *Parameterized uniform crossover* [3] is used to implement mating in BRKGAs. Let $\rho_e = 0.7$ be the probability that an offspring inherits the vector component of its elite parent. Let n denote the number of components in the solution vector of an individual. For $i = 1, \ldots, n$, the i-th component $c(i)$ of the offspring vector c takes on the value of the i-th component $e(i)$ of the elite parent e with probability ρ_e and the value of the i-th component $\bar{e}(i)$ of the non-elite parent $\bar{e}$ with probability $1 - \rho_e$.

When the next population is complete, i.e. when it has p individuals, fitness values are computed for all of the newly created random-key vectors and the population is partitioned into elite and non-elite individuals to start a new generation.

A BRKGA searches the solution space of the combinatorial optimization problem indirectly by searching the continuous n-dimensional hypercube, using the decoder

to map solutions in the hypercube to solutions in the solution space of the combinatorial optimization problem where the fitness is evaluated.

3. GRASP with path-relinking for the generalized quadratic assignment problem

A GRASP [4, 5] is a multi-start metaheuristic where at each iteration a greedy randomized solution is constructed to be used as a starting solution for local search. The best local minimum found over all GRASP iterations is output as the solution. See [6, 7, 8, 9] for recent surveys of GRASP.

GRASP iterations are independent, i.e. solutions found in previous GRASP iterations do not influence the algorithm in the current iteration. The use of previously found solutions to influence the procedure in the current iteration can be thought of as a memory mechanism. One way to incorporate memory into GRASP is with path-relinking [10, 11, 12]. In GRASP with path-relinking (GRASP+PR) [13, 14], an elite set of diverse good-quality solutions is maintained to be used during each GRASP iteration. After a solution is produced with greedy randomized construction and local search, that solution is combined with a randomly selected solution from the elite set using the path-relinking operator. The best of the combined solutions is a candidate for inclusion in the elite set and is added to the elite set if it meets quality and diversity criteria.

Mateus, Resende, and Silva [15] propose a GRASP with path-relinking heuristic for the generalized quadratic assignment problem (GQAP). In the GQAP, we are given n facilities and m locations and want to feasibly assign each facility to a location. Each facility uses a portion of the capacity of a location and each location has a fixed amount of capacity to distribute among facilities. An assignment is feasible if each location has sufficient capacity to accommodate the demands of all facilities assigned to it. Given nonnegative flows between all pairs of facilities and nonnegative distances between all pairs of locations, the GQAP seeks a feasible assignment that minimizes the sum of products of flows and distances in addition to a linear assignment component.

Algorithm 1 shows pseudo-code for the GRASP+PR heuristic proposed in [15] for the GQAP. The algorithm takes as input the set N of facilities, the set M of locations, the flow matrix A, the distance matrix B, the assignment cost matrix C, the facility demands $q_i,\ i \in N$, and the location capacities $Q_j,\ j \in M$, and outputs an assignment vector π^* specifying the location of each facility in the best solution found.

After initializing the elite set P as empty in line 1, the GRASP+PR iterations are computed in lines 2 to 24 until a stopping criterion is satisfied. During each iteration, a greedy randomized solution π' is generated in line 3. If the elite set P does not have at least ρ elements (ρ is an input parameter), then if π' is feasible and sufficiently different from all other elite set solutions, π' is added to the elite set in line 22. To define the term *sufficiently different* more precisely, let $\Delta(\pi', \pi)$ be defined as the minimum number of facility to location swaps needed to transform π' into π or vice-verse. For a given level of difference δ (δ is an input parameter), we say π' is sufficiently different from all elite solutions in P if $\Delta(\pi', \pi) > \delta$ for all $\pi \in P$, which we indicate with the notation $\pi' \not\approx P$. If the elite set P does have at least ρ elements, then the steps in lines 5 to 19 are computed.

The greedy randomized construction procedure is not guaranteed to generate a feasible solution. If the greedy randomized procedure returns an infeasible solution, a feasible solution π' is selected uniformly at random from the elite set in line 6 to be used as a surrogate for the greedy randomized solution. An approximate local search is applied using π' as a starting point in line 8, resulting in an approximate local minimum, which we denote by π'. Since elite solutions are made up of approximate local minima, then applying an approximate local search to an elite solution will, with high probability, result in a different approximate local minimum.

The approximate local search is not guaranteed to find an exact local minimum. Since π' is an approximate local minimum, the application of an approximate local search to it will, with high probability, result in a different approximate local minimum. Next, path-relinking is applied in line 10 between π' and an elite solution π^+, randomly chosen in line 9. Solution π^+ is selected with probability proportional to $\Delta(\pi', \pi^+)$. In line 11, the approximate local search is applied to π'. If the elite set is full (the maximum number of solutions in the elite set is an input parameter), then if π' is of better quality than the worst elite solution and $\pi' \not\approx P$, then it will be added to the elite set in line 14 in place of some elite solution. Among all elite solutions having cost no better than that of π', a solution π most similar to π', i.e. with the smallest $\Delta(\pi', \pi)$ value, is selected to be removed from the elite set. Ties are broken at random. Otherwise, if the elite set is not full, π' is simply added to the elite set in line 18 if $\pi' \not\approx P$.

We next summarize procedures `GreedyRandomized`, `ApproxLocalSearch`, and `PathRelinking`. These procedures are described in detail in Mateus, Resende, and Silva [15].

Procedure `GreedyRandomized` attempts to construct a greedy randomized solution to serve at a starting solution for local search. It does so by attempting, at most $\bar{t} \in [1, 100]$ times, to construct a feasible solution. In the construction process facilities and locations are selected at random with bias. To implement the randomized selection three types of probabilities are computed:

- Probability of selecting new location j: $H_j / \sum_{l \in L} H_l$, where L is the set of currently unused locations and $H_j = \sum_{l \in CL} \frac{Q_j^{h_1} Q_l^{h_2}}{b_{jl}^{h_3}}$, where Q_j is the capacity of location j, b_{jl} is the distance between locations j and l, CL is the set of previously selected locations, and input parameters h_1, h_2, h_3 are real numbers in the interval $[0, 1]$.
- Probability of selecting new facility i: $W_i / \sum_{t \in T} W_t$, where T is the subset of currently unused facilities and $W_i = q_i^{w_1} \sum_{t \in N \setminus \{i\}} a_{it}^{w_2}$, where q_i is the demand of facility i, a_{it} is the flow between facilities i and t, N is the set of facilities, and input parameters w_1, w_2 are real numbers in the interval $[0, 1]$.
- Probability of selecting a used location j: $Z_j / \sum_{r \in R} Z_r$, where R is a subset of currently used locations and $Z_j = \sum_{l \in CL \setminus \{j\}} \frac{\sigma_j^{z_1} Q_l^{z_2}}{d^{z_3} b_{jl}^{z_4}}$, where σ_j is the available capacity of location j, d is the increase in the objective function resulting from the assignment to it of the chosen facility in T, and input parameters z_1, z_2, z_3, z_4 are real numbers in the interval $[0, 1]$.

```
procedure GRASP+PR
  Data : N, M, A, B, C, q_i, Q_j.
  Result: Solution π* ∈ χ.
1  P ← ∅;
2  while stopping criterion not satisfied do
3  |  π' ← GreedyRandomized(·);
4  |  if elite set P has at least ρ elements then
5  |  |  if π' is not feasible then
6  |  |  |  Randomly select a new solution π' ∈ P;
7  |  |  end
8  |  |  π' ← ApproxLocalSearch(π');
9  |  |  Randomly select a solution π+ ∈ P;
10 |  |  π' ← PathRelinking(π', π+);
11 |  |  π' ← ApproxLocalSearch(π');
12 |  |  if elite set P is full then
13 |  |  |  if c(π') ≤ max{c(π) | π ∈ P} and π' ≉ P then
14 |  |  |  |  Replace the element most similar to π' among all
   |  |  |  |    elements with cost worst than π';
15 |  |  |  end
16 |  |  |
17 |  |  else if π' ≉ P then
18 |  |  |  P ← P ∪ {π'};
19 |  |  end
20 |  |
21 |  else if π' is feasible and π' ≉ P then
22 |  |  P ← P ∪ {π'};
23 |  end
24 end
25 return π* = min{c(π) | π ∈ P};
```

Algorithm 1: Pseudo-code for `GRASP+PR`: GRASP with path-relinking heuristic.

These probabilities are used in one of five heuristic-biased stochastic sampling schemes of Bresina [16] determined by input parameter $s \in \{1, 2, 3, 4, 5\}$. If Bresina's polynomial bias is selected, parameter $g \in \{1, 2, \ldots, 10\}$ determines the degree of the polynomial used. Consequently procedure `GreedyRandomized` takes as input 12 user-defined parameters $\bar{t}$, h_1, h_2, h_3, w_1, w_2, z_1, z_2, z_3, z_4, s, and g.

Procedure `ApproxLocalSearch` applies an approximate local search scheme from a given starting solution. Given a current solution, the method samples solutions resulting from single and double facility-to-location moves to create a set of candidate solutions (CLS) to which to move to. At each iteration the method samples at most $MaxItr$ moves, some improving, some not, and creates the set CLS with at most $MaxCLS$ elements. The method either chooses the solution π from CLS in a greedy fashion or it selects it with probability

$$\frac{G_\pi}{\sum_{\pi' \in CLS} G_{\pi'}}, \text{ where } G_{\pi'} = 1/f(\pi'),$$

where $f(\cdot)$ is the objective function the GQAP. Input parameter *CLChoice* determines which option is used. In the former case, parameters $s \in \{1, 2, 3, 4, 5\}$ and $g \in \{1, 2, \ldots, 10\}$ determine, as in construction procedure, which of Bresina's stochastic sampling schemes will be used. Consequently, procedure `ApproxLocalSearch` uses as input 6 user-defined parameters: $MaxItr \in \{1, 2, \ldots, 1000\}$, $MaxCLS \in \{1, 2, \ldots, 100\}$, $CLChoice \in \{0, 1\}$, $s \in \{1, 2, 3, 4, 5\}$, $g \in \{1, 2, \ldots, 10\}$, which determines if the greedy or which randomized selection will be used, and the real-valued $neighborhoodBalance \in [0, 1]$ which determines the proportion of single and double facility-to-location moves sampled.

Procedure `PathRelinking` takes as input 8 user-defined parameters that determine, among many options, whether forward-, backward-, or mixed-path-relinking is used, whether truncated path-relinking is used (and if so, how many steps are carried out), whether greedy or greedy randomized path-relinking is used, and the maximum number of feasibility restoration steps that can be carried out.

In addition to the above 26 parameters, the main algorithm has the following 5 parameters: the maximum size of the elite set $maxES \in \{1, 2, \ldots, 50\}$, the minimum number of elements in the elite set required for path-relinking to be used $\rho \in \{2, 3, \ldots, maxES\}$, the minimum difference parameter $\delta \in \{0, 1, \ldots, n\}$, and parameter $selectFromES \in \{0, 1\}$ which determines whether in line 9 of Algorithm 1 element π^+ is selected uniformly at random or with bias. In total, there are 30 user-defined parameters than need to be tuned.

4. Two-phase hybrid heuristic

The two-phase hybrid heuristic consists first of a tuning phase, where the BRKGA explores the GRASP+PR parameter space, followed by a solution phase, where the GRASP+PR heuristic using the parameters determined in the first phase explores the GQAP solution space seeking an optimal or near optimal assignment of facilities to locations. In the first phase, the BRKGA is run for Y_1 generations while in the second phase, the GRASP+PR heuristic is run for Y_2 iterations.

To describe the first phase of the two-phase hybrid heuristic, we first specify the encoding of the parameter-space solutions as well as the decoding of these solutions. The random-key solution vector x has $n = 30$ components, one for each tunable parameter. Each component is a random key generated in the real interval $[0, 1]$. If a parameter $i = 1, \ldots, n$ is in the real interval $[l, u]$, its random-key component $x(i)$ is decoded as $l + x(i) \cdot (u - l)$. On the other hand, if parameter i is in the discrete interval $[l, u]$, its random-key component $x(i)$ is decoded as $\lceil l - \frac{1}{2} + x(i) \cdot (u - l) \rceil$.

The fitness of a solution vector is obtained by carrying out V independent runs of the GRASP+PR heuristic using the parameters decoded from the solution vector, each run for U GRASP+PR iterations. The fitness is computed as the average objective function value of the V runs.

5. Computational results

In this section, we report on preliminary computational results with the automatic parameter tuning scheme introduced in this paper.

All experiments were done on a Dell PE1950 computer with dual quad core 2.66 GHz Intel Xeon processors and 16 Gb of memory, running Red Hat Linux nesh version 5.1.19.6 (CentOS release 5.2, kernel 2.6.18-53.1.21.el5). The two-phase BRKGA / GRASP+PR heuristic was implemented in Java and compiled

TABLE 1. Comparison of a GRASP+PR heuristic for the GQAP with manually and automatically tuned parameters. For each instance and each heuristic variant, the table lists minimum, maximum, and average times (in seconds) to find the best known solution, as well as standard deviations.

	Manually tuned				Automatically tuned			
problem	min	max	avg	sdev	min	max	avg	sdev
20-15-35	1.16	845.29	147.09	146.53	0.59	71.30	9.62	9.19
20-15-55	0.63	83.52	17.04	16.43	0.36	33.03	7.17	6.15
20-15-75	0.92	166.30	8.47	14.04	0.78	552.19	47.55	82.88
30-08-55	0.35	11.67	2.26	1.54	0.07	3.42	0.96	0.61
30-07-75	9.22	26914.03	716.08	2027.75	1.27	228.01	28.63	29.75

into bytecode with `javac` version 1.6.0_05. The random-number generator is an implementation of the Mersenne Twister algorithm [17] from the COLT[1] library.

The objective of the experiments was to compare the performance of the heuristic using the parameter obtained through manual tuning in [15] with the same heuristic using parameters automatically tuned with the BRKGA described in this paper. We consider five instances from Cordeau et al. [18]: 20-15-35, 20-15-55, 20-15-75, 30-07-75, and 30-08-55. Instance f-l-t in this class has f facilities and l locations. Parameter t controls the tightness of the constraints. The higher the value of t, the greater the tightness of the constraints. The tighter the constraints, the harder it is to find a feasible solution.

For each instance, we ran the first phase of the hybrid heuristic to automatically tune the 30 parameters of the GRASP+PR heuristic for the GQAP. The BRKGA used a population of 15 elements and ran for only 10 generations. The fitness computation was done over $V = 30$ independent runs of the GRASP+PR heuristic, each one for $U = 100$ iterations. With the automatically tuned parameters on hand, the heuristic found, in the second phase, the best solution for all five instances. In addition, 200 independent runs of the GRASP+PR heuristic (manually and automatically tuned variants) were carried out for each instance, stopping each time only after the best known solution for the instance was found. All 200 runs of each variant and for each instance found the best known solution.

The plots in Figure 1 show solutions found by the tuning procedure (on top) and the GRASP+PR with the automatically tuned parameters (on the bottom) on instance 20-15-75. As can be observed, the BRKGA finds a good parameter setting in a few generations and the GRASP+PR using these parameters quickly reaches the best known solution for the instance.

Table 1 summarizes the experiments. For each instance, the table lists statistics for both the manually and automatically tuned GRASP+PR heuristic. For each variant, the table shows the minimum, maximum, and average running times (in seconds) to find the best known solution for each instance, as well as the standard deviation computed over 200 independent runs of the GRASP+PR heuristic.

[1]COLT is a open source library for high performance scientific and technical computing in Java. See `http://acs.lbl.gov/~hoschek/colt/`.

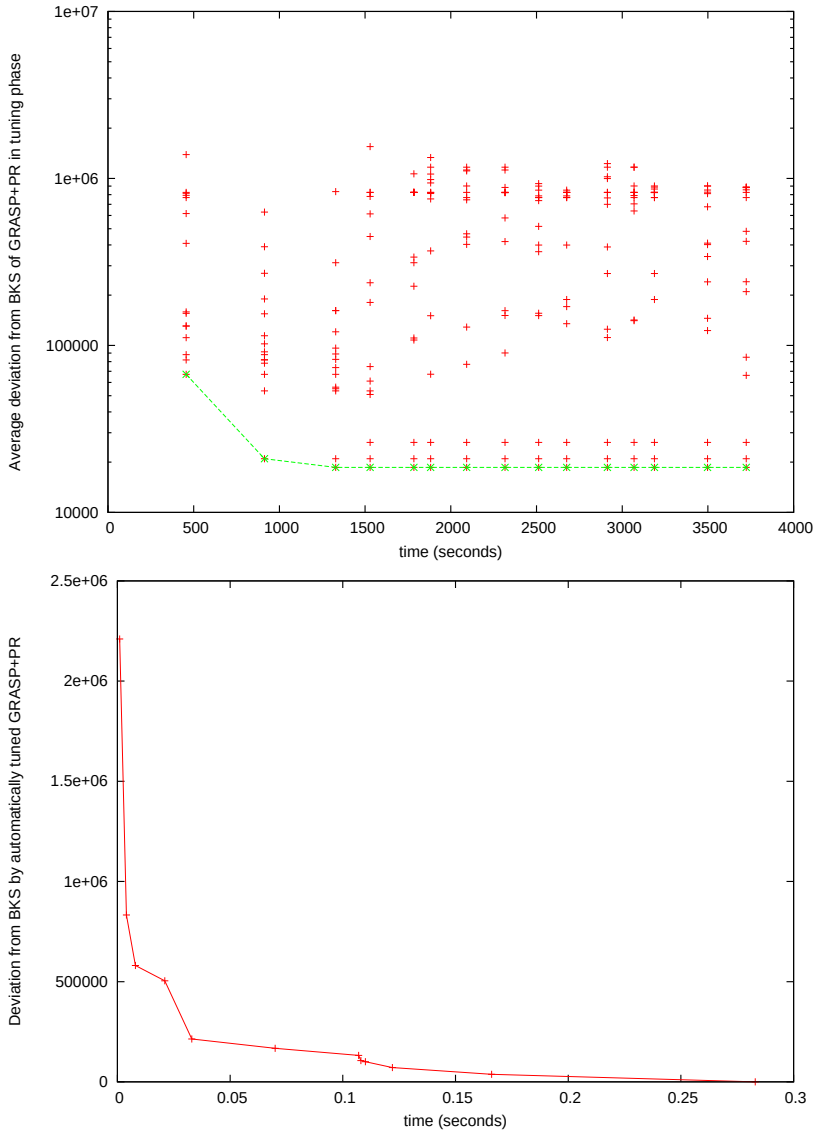

FIGURE 1. The plot on top shows, for each iteration of the BRKGA tuning procedure, the distribution of fitness values found for instance 20-15-75. The plot on the bottom shows the solutions found by the GRASP+PR heuristic using the automatically tuned parameters found by the tuning procedure. Both figures show deviations from the best known solution (BKS) for 20-15-75.

Figures 2, 3, 4, 5, and 6 show runtime distribution plots for the manually and automatically tuned GRASP+PR heuristics for, respectively, instances 20-15-35, 20-15-55, 20-15-75, 30-07-75, and 30-08-55.

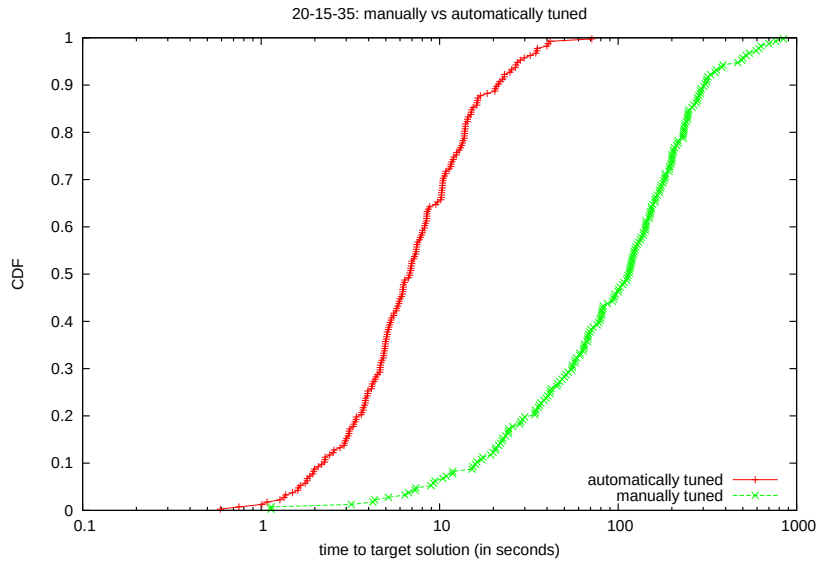

FIGURE 2. Runtime distributions for manually and automatically tuned GRASP+PR heuristics for the GQAP on instance 20-15-35. 200 independent runs of each variant were carried out and running times to find the best known solution for the instance plotted.

Times on Table 1 as well as Figures 2, 3, 4, 5, and 6 are limited to GRASP+PR and do not include the time taken by the BRKGA to automatically tune the parameters. Tuning times were, respectively, 10,739.2, 7,551.2, 3,690.3, 21,909.1, and 14,386.5 seconds for instances 20-15-35, 20-15-55, 20-15-75, 30-07-75, and 30-08-55. These times could be reduced considerably with a parallel implementation of the BRKGA as well as with the imposition of a maximum running time for the GRASP+PR heuristic run in the process of computing the fitness of the parameter settings. Poor settings often lead to configurations that struggle to find feasible assignments, thus leading to long running times. On the other hand, the times for the manually tuned heuristic do not reflect the weeks that it took for us to do the manual tuning.

The table as well as the figures clearly show that significant improvements can be obtained with automatic tuning of the parameters. On all instances except 20-15-75, the automatically tuned variant proved to find the best known solution in less time than the manually tuned variant. In the most difficult instance (30-07-75), the automatically tuned variant was on average about 25 times faster than the manually tuned variant. The ratio of maximum running times on this instance was over 118, in favor of the automatically tuned variant.

REFERENCES

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

[2] Gonçalves, J., Resende, M.: Biased random-key genetic algorithms for combinatorial optimization. Technical report, AT&T Labs Research, Florham Park, NJ 07932 (2009) (`http://www.research.att.com/~mgcr/doc/srkga.pdf`).

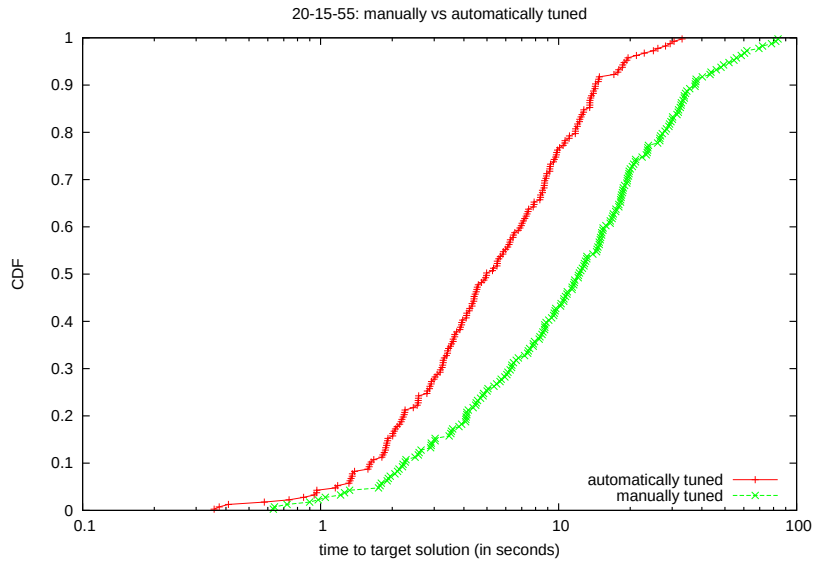

FIGURE 3. Runtime distributions for manually and automatically tuned GRASP+PR heuristics for the GQAP on instance 20-15-55. 200 independent runs of each variant were carried out and running times to find the best known solution for the instance plotted.

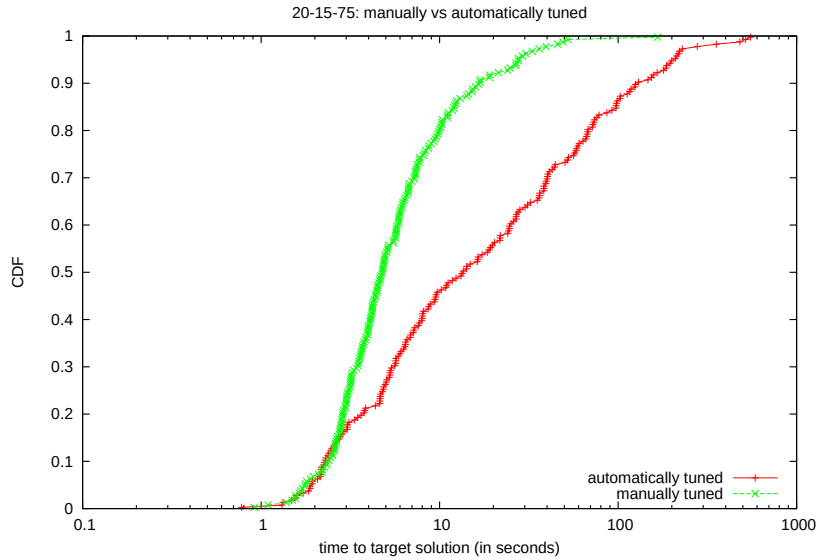

FIGURE 4. Runtime distributions for manually and automatically tuned GRASP+PR heuristics for the GQAP on instance 20-15-75. 200 independent runs of each variant were carried out and running times to find the best known solution for the instance plotted.

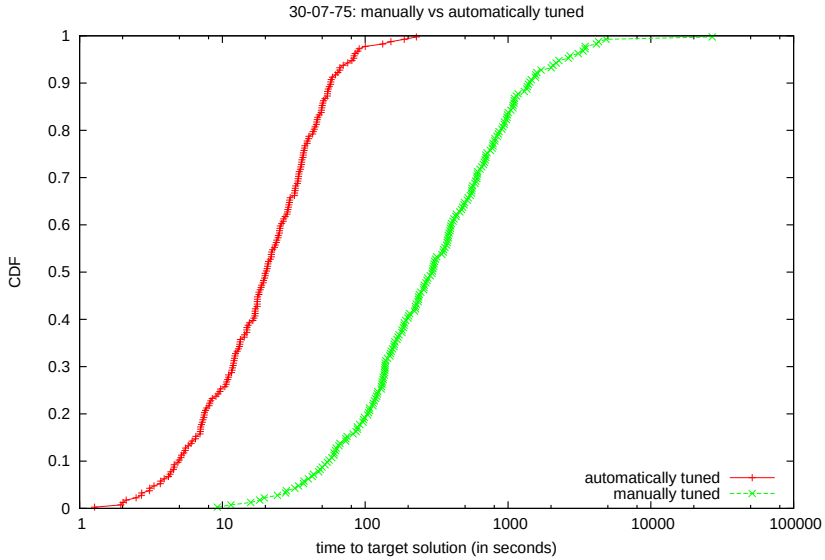

FIGURE 5. Runtime distributions for manually and automatically tuned GRASP+PR heuristics for the GQAP on instance 30-07-75. 200 independent runs of each variant were carried out and running times to find the best known solution for the instance plotted.

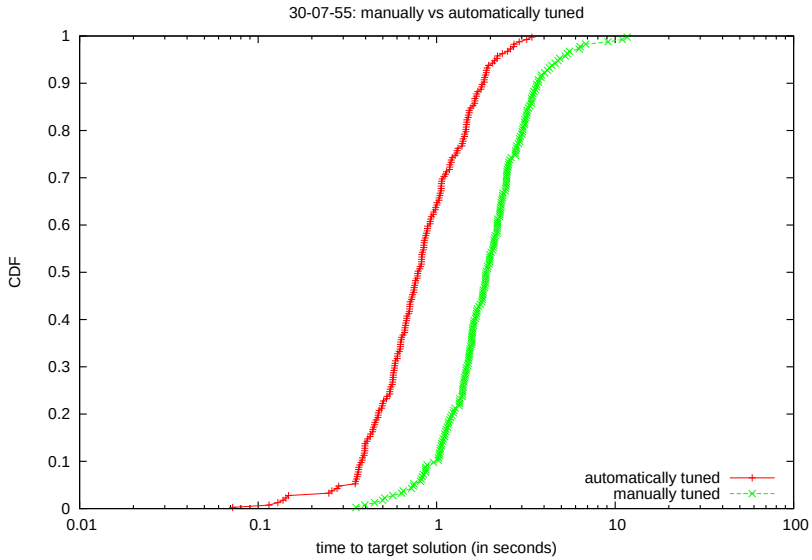

FIGURE 6. Runtime distributions for manually and automatically tuned GRASP+PR heuristics for the GQAP on instance 30-08-55. 200 independent runs of each variant were carried out and running times to find the best known solution for the instance plotted.

[3] Spears, W., DeJong, K.: On the virtues of parameterized uniform crossover. In: Proceedings of the Fourth International Conference on Genetic Algorithms. (1991) 230–236
[4] Feo, T., Resende, M.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters **8** (1989) 67–71
[5] Feo, T., Resende, M.: Greedy randomized adaptive search procedures. J. of Global Optimization **6** (1995) 109–133
[6] Resende, M., Ribeiro, C.: Greedy randomized adaptive search procedures. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers (2002) 219–249
[7] Resende, M., Ribeiro, C.: Greedy randomized adaptive search procedures: Advances and applications. In Gendreau, M., Potvin, J.Y., eds.: Handbook of Metaheuristics. 2nd edn. Springer Science+Business Media (2010)
[8] Festa, P., Resende, M.: An annotated bibliography of GRASP – Part I: Algorithms. International Transactions on Operational Research **16** (2009) 1–24
[9] Festa, P., Resende, M.: An annotated bibliography of GRASP – Part II: Applications. International Transactions on Operational Research (2009) In press.
[10] Glover, F.: Tabu search and adaptive memory programing – Advances, applications and challenges. In Barr, R., Helgason, R., Kennington, J., eds.: Interfaces in Computer Science and Operations Research. Kluwer (1996) 1–75
[11] Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics **39** (2000) 653–684
[12] Resende, M., Ribeiro, C., Glover, F., Martí, R.: Scatter search and path-relinking: Fundamentals, advances, and applications. In Gendreau, M., Potvin, J.Y., eds.: Handbook of Metaheuristics. 2nd edn. Springer Science+Business Media (2010)
[13] Laguna, M., Martí, R.: GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS Journal on Computing **11** (1999) 44–52
[14] Resende, M., Ribeiro, C.: GRASP with path-relinking: Recent advances and applications. In Ibaraki, T., Nonobe, K., Yagiura, M., eds.: Metaheuristics: Progress as Real Problem Solvers. Springer (2005) 29–63
[15] Mateus, G., Resende, M., Silva, R.: GRASP with path-relinking for the generalized quadratic assignment problem. Technical report, AT&T Labs Research Technical Report, Florham Park, NJ 07932 (2009) (`http://www.research.att.com/~mgcr/doc/gpr-gqap.pdf`).
[16] Bresina, J.: Heuristic-biased stochastic sampling. In: Proceedings of the AAAI-96. (1996) 271–278
[17] Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation **8** (1998) 3–30
[18] Cordeau, J.F., Gaudioso, M., Laporte, G., Moccia, L.: A memetic heuristic for the generalized quadratic assignment problem. INFORMS Journal on Computing **18** (2006) 433–443

(P. Festa) DMA, University of Napoli FEDERICO II, Italy.
E-mail address: `paola.festa@unina.it`

(José Fernando Gonçalves) Faculdade de Economia do Porto / NIAAD, Rua Dr. Roberto Frias, 4200-464, Porto, Portugal.
E-mail address: `jfgoncal@fep.up.pt`

(M.G.C. Resende) Internet and Network Systems Research, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.
E-mail address, M.G.C. Resende: `mgcr@research.att.com`

(R.M.A. Silva) Computational Intelligence and Optimization Group, Dept. of Computer Science, Federal University of Lavras, C.P. 3037, CEP 37200-000, Lavras, MG, Brazil
E-mail address, R.M.A. Silva: `rmas@dcc.ufla.br`