A Biased Random Key Genetic Algorithm approach for Unit Commitment Problem

Luís A.C. Roque
Dalila B.M.M. Fontes
Fernando A.C.C. Fontes

ISEP-DEMA/GECAD, Instituto Superior de Engenharia do Porto, Portugal
FEP/LIAAD-INESC Porto L.A., Universidade do Porto ,Portugal
FEUP/ISR-Porto,Universidade do Porto,Portugal
{lar@isep.ipp.pt,fontes@fep.up.pt,faf@fe.up.pt}

Abstract. *A Biased Random Key Genetic Algorithm (BRKGA) is proposed to find solutions for the unit commitment problem. In this problem, one wishes to schedule energy production on a given set of thermal generation units in order to meet energy demands at minimum cost, while satisfying a set of technological and spinning reserve constraints. In the BRKGA, solutions are encoded by using random keys, which are represented as vectors of real numbers in the interval* $[0, 1]$. *The GA proposed is a variant of the random key genetic algorithm, since bias is introduced in the parent selection procedure, as well as in the crossover strategy. Tests have been performed on benchmark large-scale power systems of up to 100 units for a* 24 *hours period. The results obtained have shown the proposed methodology to be an effective and efficient tool for finding solutions to large-scale unit commitment problems. Furthermore, from the comparisons made it can be concluded that the results produced improve upon some of the best known solutions.*

Keywords: Unit Commitment, Genetic Algorithm, Optimization, Electrical Power Generation.

1 INTRODUCTION

The Unit Commitment (UC) is a complex optimization problem well known in the power industry and adequate solutions for it have potential large economic benefits that could result from the improvement in unit scheduling. Therefore, the UC problem plays a key role in planning and operating power systems. The thermal UC problem involves scheduling the turn-on and turn-off of the thermal generating units, as well as the dispatch for each on-line unit that minimizes the operating cost for a specific time generation horizon. In addition, there are multiple technological constraints, as well as system demand and spinning reserve constraints that must be satisfied. Due to its combinatorial nature, multi-period characteristics, and nonlinearities, this problem is highly computational demanding and, thus, it is a hard optimization task solving the UC problem, specially for real-sized systems.

The methodology proposed to find solutions for this problem is a Genetic Algorithm where the solutions are encoded using random keys. A Biased Random Key Genetic Algorithm (BRKGA) is proposed to find the on/off state of the generating units for every time period as well as the amount of production of each unit at each time period.

In the past, several traditional heuristic approaches based on exact methods have been used such as dynamic programming, mixed-integer programming and benders decomposition, see e.g. [15, 6, 16]. However, more recently most of the developed methods are metaheuristics, evolutionary algorithms, and hybrids of the them, see e.g. [2, 27, 11, 7, 23, 4, 28, 1]. These latter types have, in general lead to better results than the ones obtained with the traditional heuristics. The most used metaheuristic methods are simulated annealing (SA) [20, 23], evolutionary programming (EP) [12, 19], memetic algorithm (MA)[27], particle swarm optimization (PSO) [24, 30] and genetic algorithms (GA), see e.g. [14, 25]. Comprehensive and detailed surveys can be found in [17, 21, 18].

In this paper we focus on applying Genetic Algorithms (GAs) to find good quality solutions for the UC problem. The majority of the reported GA implementations to address the UC problem are based on the binary encoding. However, studies have shown that other encoding schemes such as real valued random keys [3] can be efficient when accompanied with suitable GA operators, specially for problems where the relative order of tasks is important. In the proposed algorithm a solution is encoded as a vector of N real random keys in the interval $[0, 1]$, where N is the number of generation units. The Biased Random Key Genetic Algorithm (BRKGA) proposed in this paper is based on the framework provided by Resende and Gonçalves in [9]. BRKGAs are a variation of the Random key Genetic Algorithms (RKGAs), first introduced by Bean [3]. The bias is introduced at two different stages of the GA. On the one hand, when parents are selected we get a higher change of good solutions being chosen, since one of the parents is always taken from a subset including the best solutions. On the other hand, the crossover strategy is more likely to choose alleles from the best parent to be inherited by offspring. The work [9] provides a tutorial on the implementation and use of biased random key genetic algorithms for solving combinatorial optimization problems and many successful applications are reported in the references therein.

This paper is organized as follows. In Section 2, the UC problem is described and formulated, while in Section 3 the solution methodology proposed is explained. Section 4 describes the set of benchmark systems used in the computational experiments and reports on the results obtained. Finally, in Section 5 some conclusions are drawn.

2 UC PROBLEM FORMULATION

In the UC problem one needs to determine at each time period the turn-on and turn-off times of the power generation units, as well as the generation output subject to operational constraints, while satisfying load demands at minimum cost. Therefore, we have two types of decision variables. The binary variables, which indicate the status of each unit at each time period and the real variables, which provide the information on the amount of energy produced by each unit at each time period. The choices made must satisfy two sets of constraints: the demand constraints (regarding the load requirements

and the spinning reserve requirements) and the technical constraints (regarding generation unit constraints). The costs are made up two components: the fuel costs, i.e. production costs, and the start-up costs.

Let us now introduce the parameters and variables notation.

Decision Variables:
$\mathbf{Y_{t,j}}$: Thermal generation of unit j at time period t, in $[MW]$;
$\mathbf{u_{t,j}}$: Status of unit j at time period t (1 if the unit is on; 0 otherwise);
Auxiliary Variables:
$\mathbf{T_j^{on/off}(t)}$: Time periods for which unit j has been continuously on-line/off-line until time period t, in $[hours]$;
Parameters:
T: Number of time periods (hours) of the scheduling time horizon; **t**: Time period index;
N: Number of generation units; **j**: Generation unit index;
$\mathbf{R_t}$: System spinning reserve requirements at time period t, in $[MW]$; $\mathbf{D_t}$: Load demand at time period t, in $[MW]$;
$\mathbf{Ymin_j}$: Minimum generation limit of unit j, in $[MW]$; $\mathbf{Ymax_j}$: Maximum generation limit of unit j, in $[MW]$;
$\mathbf{N_b}$: Number of the base units; $\mathbf{T_{min,j}^{on/off}}$: Minimum uptime/downtime of unit j, in $[hours]$;
$\mathbf{T_{c,j}}$: Cold start time of unit j, in $[hours]$; $\mathbf{S_{H/C,j}}$: Hot/Cold start-up cost of unit j, in [$];
$\Delta_j^{dn/up}$: Maximum allowed output level decrease/increase in consecutive periods for unit j, in $[MW]$;

2.1 Objective Function

As already said, there are two cost components: generation costs and start-up costs. The generation costs, i.e. the fuel costs, are conventionally given by a quadratic cost function as in equation (1), while the start-up costs, that depend on the number of time periods during which the unit has been off, are given as in equation (2).

$$F_j(Y_{t,j}) = a_j \cdot (Y_{t,j})^2 + b_j \cdot Y_{t,j} + c_j, \tag{1}$$

where a_j, b_j, c_j are the cost coefficients of unit j.

$$S_{t,j} = \begin{cases} S_{H,j} & \texttt{if } T_{min,j}^{off} \leq T_j^{off}(t) \leq T_{min,j}^{off} + T_{c,j} \\ S_{C,j} & \texttt{if } T_j^{off}(t) > T_{min,j}^{off} + T_{c,j} \end{cases}, \tag{2}$$

where $S_{H,j}$ and $S_{C,j}$ are the hot and cold start-up costs of unit j, respectively.

Therefore, the cost incurred with an optimal scheduling is given by the minimization of the total costs for the whole planning period, as in equation (3).

$$\text{Minimize} \quad \sum_{t=1}^{T} \left(\sum_{j=1}^{N} \left\{ F_j(Y_{t,j}) \cdot u_{t,j} + S_{t,j} \cdot (1 - u_{t-1,j}) \cdot u_{t,j} \right\} \right). \tag{3}$$

2.2 Constraints

The constraints can be divided into two sets: the demand constraints and the technical constraints. Regarding the first set of constraints it can be further divided into load requirements and spinning reserve requirements, which can be written as follows:

1) Power Balance Constraints

The total power generated must meet the load demand, for each time period.

$$\sum_{j=1}^{N} Y_{t,j} \cdot u_{t,j} \geq D_t, t \in \{1, 2, ..., T\}. \tag{4}$$

2) Spinning Reserve Constraints
The spinning reserve is the total amount of real power generation available from on-line units net of their current production level.

$$\sum_{j=1}^{N} Ymax_j \cdot u_{t,j} \geq R_t + D_t, t \in \{1,2,...,T\}. \quad (5)$$

The second set of constrains includes unit output range, minimum number of time periods that the unit must be in each status (on-line and off-line), and the maximum output variation allowed for each unit.

3) Unit Output Range Constraints
Each unit has a maximum and minimum production capacity.

$$Ymin_j \cdot u_{t,j} \leq Y_{t,j} \leq Ymax_j \cdot u_{t,j}, \text{for } t \in \{1,2,...,T\} \text{ and } j \in \{1,2,...,N\}. \quad (6)$$

4) Ramp rate Constraints
Due to the thermal stress limitations and mechanical characteristics the output variation, levels of each online unit in two consecutive periods are restricted by ramp rate limits.

$$-\Delta_j^{dn} \leq Y_{t,j} - Y_{t-1,j} \leq \Delta_j^{up}, \text{for } t \in \{1,2,...,T\} \text{ and } j \in \{1,2,...,N\}. \quad (7)$$

5) Minimum Uptime/Downtime Constraints
The unit cannot be turned on or off instantaneously once it is committed or uncommitted. The minimum uptime/downtime constraints indicate that there will be a minimum time before it is shut-down or started-up, respectively.

$$T_j^{on}(t) \geq T_{min,j}^{on} \text{ and } T_j^{off}(t) \geq T_{min,j}^{off}, \text{for } t \in \{1,2,...,T\} \text{ and } j \in \{1,2,...,N\}. \quad (8)$$

3 METHODOLOGY

Genetic Algorithms are a global optimization technique based on natural genetics and evolution mechanisms such as survival of the fittest law, genetic recombination and selection [10, 8]. GAs provide great modeling flexibility and can easily be implemented to search for solutions of combinatorial optimization problems. Several GAs have been proposed for the unit commitment problem, see e.g. [14, 5, 26, 2, 29, 7, 1], the main differences being the representation scheme, the decoding procedure, and the solution evaluation procedure (i.e. fitness function).

Many GA operators have been used; the most common being copy, crossover, and mutation. Copy consists of simply copying the best solutions from the previous generation into the next one, with the intention of preserving the chromosomes corresponding to best solutions in the population. Crossover produces one or more *offspring* by combining the genes of solutions chosen to act as their parents. The mutation operator randomly changes one or more genes of a given chromosome in order to introduce some extra variability into the population and thus, prevent premature convergence.

The GA proposed here, i.e. the BRKGA, uses the framework proposed by Gonçalves and Resende in [9]. The algorithm evolves a population of chromosomes that are used

to assign priorities to the generation units. These chromosomes are vectors, of size N (number of units), of real numbers in the interval $[0,1]$ (called random keys). A new population is obtained by joining three subsets of solutions as follows: the first subset is obtained by copying the best solutions of the current population; the second subset is obtained by using a (biased) parameterized uniform crossover; the remaining solutions, termed mutants, are randomly generated as was the case for the initial population. The BRKGA framework is illustrated in Figure 1, which has been adapted from [9].

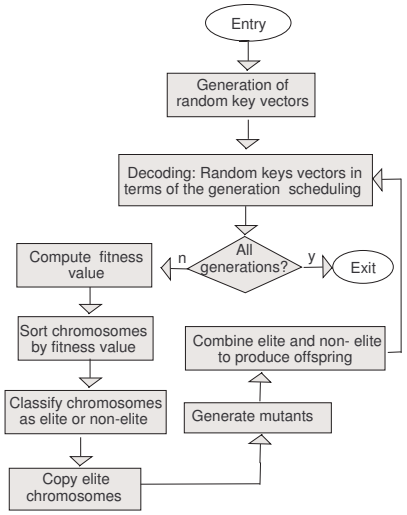

Fig. 1: The BRKGA framework.

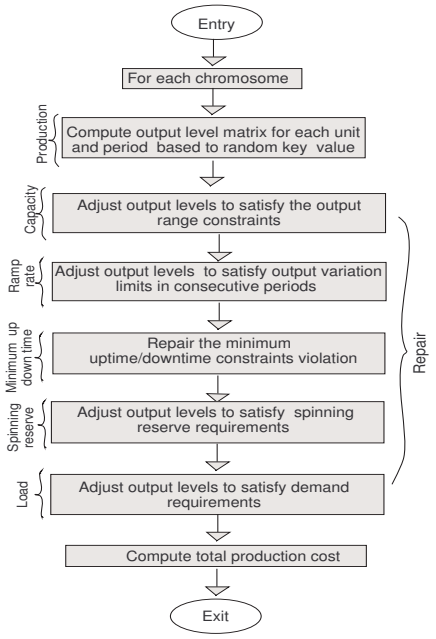

Fig. 2: Flow chart of the decoder.

Specific to our problem are the decoding procedure, as well as the fitness computation. The decoding procedure, that is how solutions are constructed once a population of chromosomes is given, is performed in two main steps, as it can be seen in Figure 2. Firstly, a solution satisfying the load demand, for each period is obtained. In this solution, the units production is proportional to their priority, which is given by the random key value. Then, these solutions are checked for constraints satisfaction.

3.1 Decoding Procedure

Given a vector of numbers in the interval $[0,1]$, say $RK = (r_1, r_2, ..., r_N)$, percent vectors $V' = \left(v'_1, v'_2, ..., v'_{N_b}\right)$, $V = (v_1, v_2, ..., v_N)$ and rank vector $O = (o_1, o_2, ..., o_N)$ are computed. Each element v_j is computed as $v_j = \frac{r_j}{\sum_i^N r_i}$, $j = 1, 2, ..., N$ and v'_j is given

by $v'_j = \frac{r_j}{\sum_i^{N_b} r_i}, j = 1,2,...,N_b$, where N_b is the number of base units, while each o_i is defined taking into account the descending order of the *RK* value.

Then an output generation matrix Y is obtained, where each element $Y_{t,j}$ gives the production level of unit $j = o_i, i = 1,...,N$ for time period t and is computed as the product of the percentage vectors V' or V by the periods demand D_t, as illustrated in the following pseudocode:

Algorithm 1 Initial matrix generation output

$i = 1$
$d = D_t$
while $i \leq N$ and $d > 0$ **do**
  $j = o_i$
  if $j \leq N_b$ **then**
    if $D_t + R_t \leq \sum_{k=1}^{N_b} Ymax_k$ **then**
      $Y_{t,j} = v'_j.D_t$
      $d = d - Y_{t,j}$
    else $\{D_t + R_t > \sum_{k=1}^{N_b} Ymax_k\}$
      $Y_{t,j} = Ymax_j$
      $d = d - Y_{t,j}$
    end if
  else $\{j > N_b\}$
    if $D_t + R_t \leq \sum_{k=1}^{N_b} Ymax_k$ **then**
      $Y_{t,j} = 0$
    else $\{D_t + R_t > \sum_{k=1}^{N_b} Ymax_k\}$
      if $d > Ymax_j$ **then**
        $Y_{t,j} = v_j.D_t$
        $d = d - Y_{t,j}$
      else $\{d < Ymax_j\}$
        $Y_{t,j} = d$
        $d = 0$
      end if
    end if
  end if
  Next i
end while

The production level of unit j for each time period t however, may not be admissible and therefore, the solution obtained may be unfeasible. Hence, the decoding procedure also incorporates a repair mechanism. This mechanism forces constraints satisfaction.

The repair mechanism starts by forcing the output level of each unit to be in its output range, as given in equation (9).

$$Y_{t,j} = \begin{cases} Ymax_j & \text{if } Y_{t,j} > Ymax_j \\ Y_{t,j} & \text{if } Ymin_j \leq Y_{t,j} \leq Ymax_j \\ Ymin_j & \text{if } \chi \cdot Ymin_j \leq Y_{t,j} < Ymin_j \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

where $\chi \in [0,1]$ is a scaling factor.

At the same time that the ramp constrains are ensured for a specific time period t, new output limits ($Y_{t,j}^{max}$ and $Y_{t,j}^{min}$ upper and lower limits, respectively) must be imposed, for the following period $t+1$, since their value depends on the output level of the current period t. Equations (10) and (11) show how this is done.

$$Y_{t,j} = \begin{cases} Y_{t,j}^{max} & \text{if } Y_{t,j} \geq Y_{t,j}^{max} \\ Y_{t,j} & \text{if } Y_{t,j}^{min} < Y_{t,j} < Y_{t,j}^{max} \\ Y_{t,j}^{min} & \text{if } \mu \cdot Y_{t,j}^{min} \leq Y_{t,j} \leq Y_{t,j}^{min} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $Y_{1,j}^{max} = Ymax_j$, $Y_{1,j}^{min} = Ymin_j$ and

$$Y_{t,j}^{max} = min\left\{Ymax_j, Y_{t-1,j} + \Delta_j^{up}\right\}, Y_{t,j}^{min} = max\left\{Ymin_j, Y_{t-1,j} - \Delta_j^{dn}\right\}. \tag{11}$$

After ensuring the unit output range constraints and the ramp rate constraints, it is still needed to guarantee that minimum up/down time constraints are satisfied. The adjustment of the unit status can be obtained using the repair mechanism illustrated in Figure 3. As it can be seen, for two consecutive periods the unit status can only be changed if the $T_{min}^{on/off}$ is already satisfied, for a previously turned on or off unit, respectively.

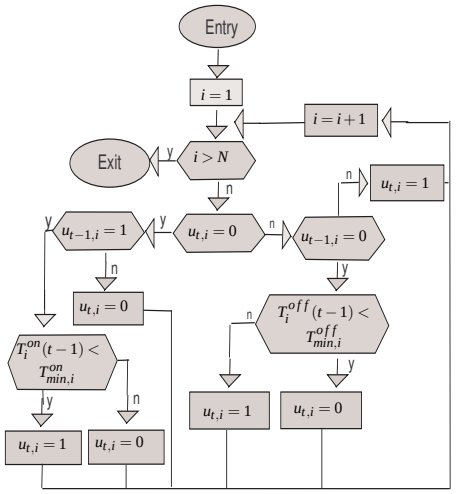

Fig. 3: Minimum up down time repair mechanism.

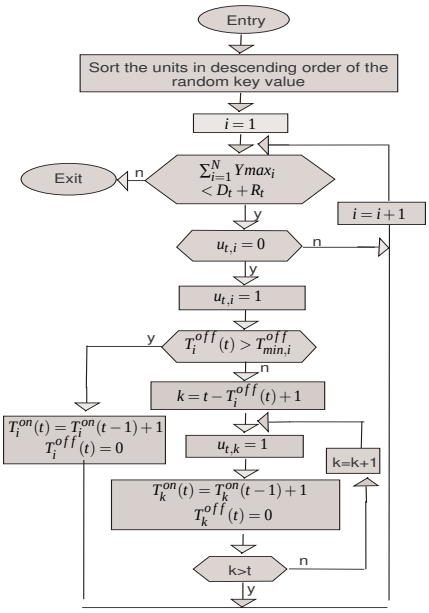

Fig. 4: Handling spinning reserve constraint.

For each period, it may happen that the spinning reserve requirement is not satisfied. If the number of on-line units is not enough, some off-line units will be turned on, one at the time, until the cumulative capacity matches or is larger than $D_t + R_t$ as shown in Figure 4. In doing so, units are considered in descending order of priority, i.e. random key value. After ensuring the spinning reserve satisfaction, it may happen that we end up with excessive spinning reserve. Since this is not desirable due to additional operational costs involved, we tried to decommit some units. Then units are considered for turning off-line, in ascending order of priority until their cumulative capacity reaches, $D_t + R_t$.

At the end of this procedure we have found the u and Y matrices specifying which units are being operated at each time period and how much its one is producing. However, it may happen that the production matches, is larger than, or lesser than the demand. In order to compute the total cost of the current solution we first must find how much each unit is producing.

If there is excessive production, the on-line units production is decreased to its minimum allowed value, one at the time, until either all are set to the minimum production or the production reaches the load demand value. In doing so, units are considered in descending order of priority, i.e. random key value. It should be notice that by reducing production at time period t the production limits at time period $t+1$ change, and the new values must be respected. Therefore, the minimum allowed production is given by $max\left\{Y_{t,j}^{min}, Y_{t+1,j} - \Delta_j^{up}\right\}$. This is repeated no more than N times. If there is lack of production, the on-line units production is increased to its maximum allowed value, one at the time, until either all are set to the maximum production or the production reaches the load demand value. In doing so, units are considered in ascending order of priority, i.e. random key value. It should be noticed that by increasing production at time period t the production limits at time period $t+1$ change, and the new values must be respected. Therefore, the maximum allowed production is given by $min\left\{Y_{t,j}^{max}, Y_{t+1,j} + \Delta_j^{dn}\right\}$. Again, this is repeated no more than N times. Once these repairing stages have been performed, the solutions obtained are feasible and the respective total cost is computed.

3.2 GA Configuration

The current population of solutions is evolved by the GA operators onto a new population as follows:

- 20% of the best solutions (elite set) of the current population are copied;
- 20% of the new population is obtained by introducing mutants, that is by randomly generating new sequences of random keys, which are then decoded to obtain mutant solutions. Since they are generated using the same distribution as the original population, no genetic material of the current population is brought in;
- Finally, the remaining 60% of the population is obtained by biased reproduction, which is accomplished by having both a biased selection and a biased crossover.

The selection is biased since, one of the parents is randomly selected from the elite set of solutions (of the current population), while the other is randomly selected from the remainder solutions. This way, elite solutions are given a higher chance of mating,

and therefore of passing on their characteristics to future populations. Genes are chosen by using a biased uniform crossover, that is, for each gene a biased coin is tossed to decide on which parent the gene is taken from. This way, the offspring inherits the genes from the elite parent with higher probability (0.7 in our case).

4 NUMERICAL RESULTS

A set of benchmark systems has been used for the evaluation of the proposed algorithm. Each of the problems in the set considers a scheduling period of 24 hours. The set of systems comprises six systems with 10 up to 100 units. A base case with 10 units was initially defined, and the others have been obtained by considering copies of these units. The base 10 units system and corresponding 24 hours load demand are given in [14]. To generate the 20 units problem, the 10 original units have been duplicated and the load demand doubled. An analogous procedure was used to obtain the problems with 40, 60, 80, and 100 units. In all cases, the spinning reserve requirements were set to 10% of the load demand.

Several computational experiments were made in order to choose the parameter values. The BRKGA was implemented with biased crossover probability as main control parameter. The parameter ranges used in our experiments were $0.5 \leq P_c \leq 0.8$ with step size 0.1 which gives 4 possible values for biased crossover probability. The results obtained have shown no major differences. Nevertheless, the results reported here refer to the best obtained ones, for which the number of generations was set to $10N$, the population size was set to $min\{2N, 60\}$, the biased crossover probability was set to 0.7, and the scaling factor $\chi = 0.4$. Due to the stochastic nature of the BRKGA, each problem was solved 20 times. We compare the results obtained with the best results reported in literature. In tables 1, 2, and 3, we compare the best, average, and worst results obtained, for each of the six problems, with the best of each available in literature. As it can be seen, for two out of the six problems solved our best results improve upon the best known results, while for the other four it is within 0.02% and 0.18% of the best known solutions. Moreover when our algorithm is not the best, it is the second best.

Table 1: Comparison between best results obtained by the BRKGA and the best ones reported in literature.

Size	GA	SA	LRGA	EP	IPSO	BRKGA	Ratio	**BRKGA rank**
10	563977	565828	564800	564551	**563954**	563977	100	2nd
20	1125516	1126251	**1122622**	1125494	1125279	1124470	100.16	2nd
40	2249715	2250063	**2242178**	2249093	2248163	2246287	100.18	2nd
60	3375065	–	3371079	3371611	**3370979**	3368542	99.93	1st
80	4505614	4498076	4501844	4498479	**4495032**	4493658	99.97	1st
100	5626514	5617876	**5613127**	5623885	5619284	5614522	100.02	2nd

For each type of solution presented (best, average, and worst) we compare each single result with the best respective one (given in bold) that we were able to find in the literature. The results used have been taken from a number of works as follows: SA [23], LRGA [5], SM[27], GA [22], EP[13] and IPSO[30].

Table 2: Comparison between average results obtained by the BRKGA and the best averages reported in literature.

Size	SA	SM	BRKGA	Ratio	**BRKGA rank**
10	**565988**	566787	563996	99.65	1st
20	**1127955**	1128213	1124753	99.72	1st
40	2252125	**2249589**	2247534	99.91	1st
60	–	**3394830**	3372104	99.33	1st
80	4501156	**4494378**	4495632	100.03	2nd
100	5624301	**5616699**	5616734	100.00	2nd

Table 3: Comparison between worst results obtained by the BRKGA and the best worst ones reported in literature.

Size	GA	SA	SM	EP	IPSO	BRKGA	Ratio	**BRKGA rank**
10	565606	566260	567022	566231	**564579**	564028	99.90	1st
20	1128790	1129112	1128403	1129793	**1127643**	1125671	99.83	1st
40	2256824	2254539	**2249589**	2256085	2252117	2248510	99.95	1st
60	3382886	–	3408275	3381012	**3379125**	3379915	100.02	2nd
80	4527847	4503987	**4494439**	4512739	4508943	4499207	100.11	2nd
100	5646529	5628506	**5616900**	5639148	5633021	5619581	100.05	2nd

Another important feature of the proposed algorithm is that, as it can be seen in Table 4, the variability of the results is very small. The difference between the worst and best solutions found for each problem is always below 0.3%, while if the best and the average solutions are compared this difference is never larger than 0.11%. This allows for inferring the robustness of the solution since the gaps between the best and the worst solutions are very small. Furthermore our worst solutions, when worse than the best worst solutions reported are always within 0.11% of the latter, see Table 3. This is very important since the industry is reluctant to use methods with high variability as this may lead to poor solutions being used.

Table 4: Analysis of the variability of the results and execution time.

Size	Best	Average	Worst	$\frac{Av-Best}{Best}\%$	$\frac{Worst-Best}{Best}\%$	St. deviation(%)	Av.Time(s)
10	563977	563996	564028	0.003	0.09	0.003	5.1
20	1124470	1124753	1125671	0.03	0.11	0.03	19.8
40	2246287	2247534	2248510	0.06	0.1	0.08	86.6
60	3368542	3372104	3379915	0.11	0.3	0.12	198.0
80	4493658	4495632	4499207	0.04	0.12	0.06	343.8
100	5614522	5616734	5619581	0.04	0.09	0.05	534.3

The BRKGA has been implemented on Matlab and executed on a Pentium IV Core Duo personal computer T 5200, 1.60GHz and 2.0GB RAM. In table 4 we can see the

scaling of the execution time with the system size for the proposed BRKGA. Regarding the computational time no exact comparisons may be done since the values are obtained on different hardware, and on the other hand, some authors only report their computational times graphically, as is the case in [30] [1]. However, in Figure 5 it can easily be seen that ours are about the same magnitude of the best execution times as in SA [23].

5 CONCLUSION

A methodology based on a Biased Random Key Genetic Algorithm, following the ideas presented in [9], for finding solutions to the unit commitment problem has been presented. In the solution methodology proposed real valued random keys are used to encode solutions, since they have been proven to perform well in problems where the relative order of tasks is important.

The proposed algorithm was applied to systems with 10, 20, 40, 60, 80, and 100 units with a scheduling horizon of 24 hours. The numerical results have shown the proposed method to improve upon current state of the art, since only for three problems it was not capable of finding better solutions. Furthermore, the results show a further very important feature, lower variability. It should be notice that the difference between the worst and best solutions is always below 0.30%, while the difference between the best and the average solutions is always below 0.11%. This is very important since methods to be used in industrial applications are required to be robust, therefore preventing the use of very low quality solutions.

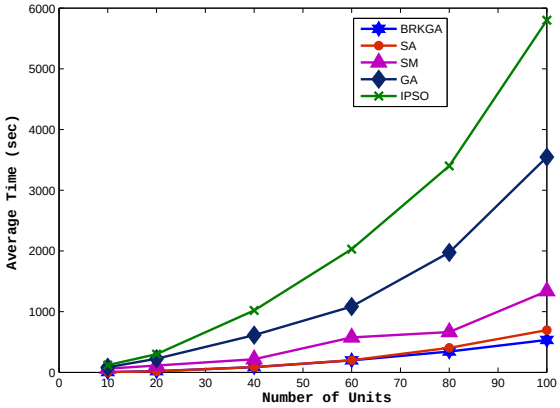

Fig. 5: Average execution time of the different methods.

[1] The computational times of the IPSO in Figure 5 are estimated from the results reported graphically.

Acknowledgments. *The financial support by FCT, POCI 2010 and FEDER, through project PTDC/EGE-GES/099741/2008 is gratefully acknowledged.*

References

1. K. Abookazemi, M.W. Mustafa, and H. Ahmad. Structured Genetic Algorithm Technique for Unit Commitment Problem. *International Journal of Recent Trends in Engineering*, 1(3):135–139, 2009.
2. J. M. Arroyo and A.J. Conejo. A parallel repair genetic algorithm to solve the unit commitment problem. *IEEE Transactions on Power Systems*, 17:1216–1224, 2002.
3. J.C. Bean. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*, 6(2), 1994.
4. Y.M. Chen and W.S. Wang. Fast solution technique for unit commitment by particle swarm optimisation and genetic algorithm. *International Journal of Energy Technology and Policy*, 5(4):440–456, 2007.
5. C.P. Cheng, C.W. Liu, and G.C. Liu. Unit commitment by Lagrangian relaxation and genetic algorithms. *IEEE Transactions on Power Systems*, 15:707–714, 2000.
6. A. I. Cohen and M. Yoshimura. A Branch-and-Bound Algorithm for Unit Commitment. *IEEE Transactions on Power Apparatus and Systems*, 102:444–451, 1983.
7. G. Dudek. Unit commitment by genetic algorithm with specialized search operators. *Electric Power Systems Research*, 72(3):299–308, 2004.
8. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York / USA, Addison-Wesley, 1989.
9. J.F. Gonçalves and M.G.C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 2010. Published online 27 August 2010. DOI: 10.1007/s10732-010-9143-1.
10. J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.
11. L. Jenkins and G.K. Purushothama. Simulated annealing with local search-a hybrid algorithm for unit commitment. *IEEE Transactions on Power Systems*, 18(1):1218–1225, 2003.
12. K. Juste. Evolutionary programming solution to the unit commitment problem. *IEEE Transactions on Power Systems*, 14(4):1452–1459, 1999.
13. K.A. Juste, H. Kita, E. Tanaka, and J. Hasegawa. An evolutionary programming solution to the unit commitment problem. *IEEE Transactions on Power Systems*, 14(4):1452–1459, 1999.
14. S.A. Kazarlis, A.G. Bakirtzis, and V. Petridis. A Genetic Algorithm Solution to the Unit Commitment Problem. *IEEE Transactions on Power Systems*, 11:83–92, 1996.
15. F.N. Lee. Short-term unit commitment-a new method. *IEEE Transactions on Power Systems*, 3(2):421–428, 1998.
16. A. Merlin and P. Sandrin. A new method for unit commitment at Electricit de France. *IEEE Transactions on Power Apparatus Systems*, 2(3):1218–1225, 1983.
17. N.P. Padhy. Unit commitment using hybrid models: a comparative study for dynamic programming, expert system, fuzzy system and genetic algorithms. *International Journal of Electrical Power & Energy Systems*, 23(8):827–836, 2001.
18. I. J. Raglend and N. P. Padhy. Comparison of Practical Unit Commitment Solutions. *Electric Power Components and Systems*, 36(8):844–863, 2008.
19. C.C.A. Rajan and MR Mohan. An evolutionary programming-based tabu search method for solving the unit commitment problem. *Power Systems, IEEE Transactions on*, 19(1):577–585, 2004.
20. CCA Rajan, MR Mohan, and K. Manivannan. Refined simulated annealing method for solving unit commitment problem. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 1, pages 333–338. IEEE, 2002.
21. S. Salam. Unit commitment solution methods. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 26, pages 600–605, 2007.
22. T. Senjyu, H. Yamashiro, K. Uezato, and T. Funabashi. A unit commitment problem by using genetic algorithm based on unit characteristic classification. In *IEEE Power Engineering Society Winter Meeting, 2002*, volume 1, 2002.
23. D.N Simopoulos, S.D. Kavatza, and C.D. Vournas. Unit commitment by an enhanced simulated annealing algorithm. *IEEE Transactions on Power Systems*, 21(1):68–76, 2006.
24. P. Sriyanyong and YH Song. Unit commitment using particle swarm optimization combined with Lagrange relaxation. In *Power Engineering Society General Meeting, 2005. IEEE*, pages 2752–2759. IEEE, 2005.
25. KS Swarup and S. Yamashiro. Unit commitment solution methodology using genetic algorithm. *Power Systems, IEEE Transactions on*, 17(1):87–91, 2002.
26. K.S. Swarup and S. Yamashiro. Unit Commitment Solution Methodology Using Genetic Algorithm. *IEEE Transactions on Power Systems*, 17:87–91, 2002.
27. J. Valenzuela and A.E. Smith. A seeded memetic algorithm for large unit commitment problems. *Journal of Heuristics*, 8(2):173–195, 2002.
28. A. Viana, J.P. Sousa, and M.A. Matos. Fast solutions for UC problems by a new metaheuristic approach. *IEEE Electric Power Systems Research*, 78:1385–1389, 2008.
29. W. Xing and F.F. Wu. Genetic algorithm based unit commitment with energy contracts. *International Journal of Electrical Power & Energy Systems*, 24(5):329–336, 2002.
30. B. Zhao, CX Guo, BR Bai, and YJ Cao. An improved particle swarm optimization algorithm for unit commitment. *International Journal of Electrical Power & Energy Systems*, 28(7):482–490, 2006.