An Improved Biased Random Key Genetic Algorithm Approach to the Unit Commitment Problem[☆]

L. A. C. Roque[a], D.B.M.M. Fontes[b], F. A. C. C. Fontes[c]

[a]*DEMA, Instituto Superior de Engenharia do Porto, 4200-072 Porto, Portugal*
[b]*LIAAD-INESC-TEC, Faculdade de Economia, Universidade do Porto, 4200-464 Porto, Portugal*
[c]*ISR-Porto, Faculdade de Engenharia, Universidade do Porto 4200-465 Porto, Portugal*

Abstract

In this work, a Biased Random Key Genetic Algorithm (BRKGA) is proposed to address the Unit Commitment (UC) problem. In the UC problem, one wishes to schedule a subset of a given set of thermal generation units and also to determine their production output in order to meet energy demands at minimum cost. In addition, the solution must satisfy a set of technological and spinning reserve constraints. The main motivation for choosing a BRKGA is its reported good performance on many combinatorial optimization problems.

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 with 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 UC problems. Furthermore, from the comparisons made it can be concluded that the results produced improve upon the solutions obtained by reported state-of-the-art methodologies.

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

[☆]We acknowledge support of ERDF (FEDER) and COMPETE through FCT within projects PTDC/EGE-GES/099741/2008 and PTDC/EEA-CRO/116014/2009.

Email addresses: lar@isep.ipp.pt (L. A. C. Roque), fontes@fep.up.pt (D.B.M.M. Fontes), faf@fe.up.pt (F. A. C. C. Fontes)

1. INTRODUCTION

The Unit Commitment (UC) problem plays a key role in planning and operating power systems since its optimal scheduling not only have the potential of saving millions of dollars, but also of maintaining system reliability by keeping a proper spinning reserve [26]. The UC problem is an optimization problem where one wishes to determine the on/off status of the generation units at minimum operating costs. In addition, the production of the committed units, which also has to be determined, must be such that it satisfies demand and spinning reserve constraints. Furthermore, a large set of technological constraints are also imposed on generation units. Due to its combinatorial nature, multi-period characteristics, and nonlinearities, this problem is highly computational demanding and, thus, it is a hard optimization task to solve it for real sized systems. The UC problem has been extensively studied in the literature. Several methodologies, based on exact and on approximate algorithms have been reported. Optimal solutions can only be obtained for small sized problem instances, through the solutions of the corresponding Mixed Integer Quadratic Programming (MIQP) model.

In the past, several traditional heuristic approaches have been proposed, based on exact methods such as Dynamic Programming, Branch and Bound, Lagrangian Relaxation, and other for Mixed-Integer Programming, see e.g. [23, 5, 16, 28, 9, 10, 30]. Most of the recently developed methods are metaheuristics, evolutionary algorithms, and hybrids of the them, see e.g. [39, 35, 6, 17, 20, 29, 15]. These latter types have, in general, lead to better results than the ones obtained with the traditional heuristics.

In this paper, a Biased Random Key Genetic Algorithm (BRKGA) is proposed to address the UC problem. This work is an improved version of [29], since in addition to incorporating a local search procedure, it also has improved decoding and repair procedures. The BRKGA proposed here is based on the framework provided by Gonçalves and Resende in [14], which has been used in other important applications in an effective and efficient way [12, 33, 13, 11, 34]. BRKGAs are a variation of the random key genetic algorithms, first introduced by Bean [4]. The bias is introduced at two different stages of the GA. On the one hand, when parents are selected, good solutions have a higher chance of 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. In our BRKGA we also include repair mechanisms and thus, all the individuals considered for evaluation are feasible. The main reasons for using repair mechanisms are 1) to work on bounded

search spaces (consisting of only feasible solutions) and 2) to avoid the problem of choosing penalties of different nature for each of the violated constraints [22].

The BRKGA is capable of finding better solutions than the best ones currently known for most of the benchmark problems solved. Furthermore, the computational time requirements are modest and similar to those of other recent approaches.

2. UC PROBLEM FORMULATION

The Unit Commitment problem consists of a set of generation units for which one needs to decide when each unit is turned on or turned off along a predefined time horizon. In addition, for each time period and each turned on generation unit one needs to decide on how much it should be producing. Before giving the mathematical formulation, we introduce the parameters and variables used.

Indexes:	$\mathbf{t}$: Time period index; $\mathbf{j}$: Generation unit index;
Decision Variables:	$\mathbf{y_{t,j}}$: Thermal generation of unit j at time t, in $[MW]$; $\mathbf{u_{t,j}}$: Status of unit j at time t (1 if it is on; 0 otherwise);
Auxiliary Variables:	$\mathbf{T_j^{on/off}(t)}$: Number of time periods for which unit j has been continuously on-line/off-line until time t, in $[hours]$;
Parameters:	$\mathbf{T}$: Number of time periods (hours) of the scheduling time horizon; $\mathbf{N}$: Number of generation units; $\mathbf{R_t}$: System spinning reserve requirements at time 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 $[MW]$; $\mathbf{T_{c,j}}$: Cold start time of unit j, in $[hours]$; $\mathbf{T_{min,j}^{on/off}}$: Minimum uptime/downtime 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 $[MW]$;

The model has two types of decision variables. Binary decision variables $u_{t,j}$, which are either set to 1, meaning that unit j is committed at time period t; or otherwise are set to zero. Real valued variables $y_{t,j}$, which indicate the amount of energy produced by unit j at time period t. Such decisions are limited by two types of constraints: load constraints, consisting of demand and spinning reserve constraints; and technological constraints. The objective of the UC problem is the minimization of the total operating costs over the scheduling horizon.

2.1. Objective Function

The objective function is made of 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 follows:

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

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

The start-up costs, that depend on the number of time periods during which the unit has been off, are given by

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

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,

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

2.2. Constraints

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

1) Power Balance Constraints: The total power generated must meet the load demand, for each time period.

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

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.

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

The second set of constrains includes limits on the unit output range, the maximum output variation allowed for each unit (ramp rate constraints) and the minimum number of time periods that the unit must be continuously in each status

(on-line or off-line).

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

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

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

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

5) Minimum Uptime/Downtime Constraints: The unit cannot be turned on or turned off instantaneously once it is committed or decommitted. The minimum uptime/downtime constraints impose a minimum number of time periods that must elapse before the unit can change its status.

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

3. PREVIOUS METHODOLOGIES ADDRESSING THE UC PROBLEM

In this section, we start by describing several traditional heuristic approaches based on exact methods that, in the past, have been reported in literature. Then, we describe methods based on metaheuristics, mainly evolutionary algorithms, and hybrids of the them, which more recently have been reported in the literature.

Dynamic Programming (DP) was the earliest optimization-based method to be applied to the UC problem. The advantage of DP is its ability to maintain solution feasibility. The disadvantage is the curse of dimensionality, which may result in unacceptable computational time. Due to the enumerative nature of the dynamic programming, it suffers from a long processing time that expands exponentially with the size of the problem. Thus, only small sized problems can be solved. Therefore, in practice many heuristic strategies have been introduced to limit the dynamic search for a large system. The most widely used method to reduce the dimension is based on a priority list. The list is typically formed by ranking the units based on their marginal power production cost or average full load cost index [32]. More recently, other approximate methods based on DP have been proposed for the UC problem and its variants. For instance in [28] it is proposed a DP algorithm based on linear relaxation of the on/off status of the units and on sequential commitment of units one by one for the UC in multi-period combined heat and

power production planning under the deregulated power market. In [30] a DP technique with a fuzzy and simulated annealing based unit selection procedure has been proposed. The computational requirements are reduced by minimizing the number of prospective solution paths to be stored at each stage of the search procedure through the use of heuristics, such as priority ordering of the units, unit grouping, fast economic dispatch based on priority ordering, avoidance of repeated economic dispatch. Nonetheless, the results produced are not amongst the best performing methods found in the literature. Not many works on the UC problem make used of Branch-and-Bound (BB). In the earlier ones [21, 5] the authors address the UC problem with time-dependent start-up costs, demand and reserve constraints and minimum up and down time constraints. However the authors do not incorporate ramp rate constraints. Furthermore, in [5] it is considered that the fuel consumption is given by a linear cost function, which constitutes another major drawback. In [16] a two-phase procedure is proposed. In the first phase, through the constraint satisfaction techniques the constraints are propagated as much as possible to reduce the search domain. The second phase fulfills the economic dispatch function on the committed units, obtaining an upper bound. The weakness of BB methods is the exponential growth in the computational time with problem dimension. Lagrangian Relaxation (LR) is capable of solving large scale UC problems in a fast manner, however the solutions obtained are, usually, suboptimal. Based on the LR approach, the UC problem can be written in terms of 1) a cost function that is the sum of terms each involving a single unit, 2) a set of constraints involving a single unit, and 3) a set of coupling constraints involving all of the units (the generation and reserve constraints), one for each hour in the study period. An approximate solution to this problem can be obtained by adjoining the coupling constraints onto the cost function by using Lagrange multipliers. The resulting relaxed problem is to minimize the so-called Lagrangian subject to the unit constraints. LR was first applied to solve the UC problem in [23], without considering ramp constraints. In [3] LR is used to disaggregate the model into separate subproblems, one for each unit. The author tests the method on a 10-units system with exponential start-up costs, see case study 5. Recently, in [9] an effective Lagrangian relaxation approach for the UC problem has been proposed. This approach relies on an exact algorithm for solving the single-unit commitment problem (proposed in [8]). More recently, in [7] two lagrangian relaxation methods are proposed: one based on subgradient optimization and the other based on cutting planes. They were tested on several problem instances generated by the authors with a simpler and linear cost function, but not on the usual benchmark ones. Therefore, no comparisons with alternative methods are

possible. From the tests performed, it was concluded that the subgradient method yields better results. By solving the MIQP model, optimal solutions can be found, but the computational time requirements are enormous and, usually, increase exponentially with the problem size, even with the availability of efficient software packages (such as CPLEX, LINDO), as will be seen in the results section. Some authors have tried to improve on the MIQP by reformulating the UC problem as a mixed integer linear programming problem by means of piece-wise linear approximations of the cost function, see e.g., [9, 10].

Regarding methods based on metaheuristics, there is recent literature reporting results on evolutionary programming [18], particle swarm optimization [39], quantum evolutionary algorithms [17, 20], memetic algorithms [37], and genetic algorithms [19, 2, 35, 6, 29]. Juste [18] employs evolutionary programming in which populations of individuals are evolved through random changes, competition, and selection. The UC schedule is coded as a string of symbols and viewed as a candidate for reproduction. Initial populations of such candidates are randomly produced to form the basis of subsequent generations. Zhao et al. [39] introduce an improved particle swarm optimization (IPSO) with adoption of the orthogonal design for generating the initial population scattered uniformly over a feasible solution space. This method has been tested on the problems of case study 1 with good results, recently outperformed by [17, 20]. In these works, Quantum-inspired Evolutionary Algorithms (QEA) are proposed. The QEA method is based on the concept and principles of quantum computing, such as quantum bits, quantum gates and superposition of states. QEA employs quantum bit representation, which has better population diversity compared to other representations used in evolutionary algorithms, and uses quantum gate to drive the population towards the best solution. The mechanism of QEA can inherently treat the balance between exploration and exploitation, thus incorporating a sort of local search. In [17, 20] works the problem is divided into two subproblems: 1) schedule on/off status of the units and 2) determine power outputs of the committed units. In both works, repair mechanisms are used to accelerate the solution quality and to ensure that unit schedules generated by QEA are feasible. In [17] the conventional QEA is improved by introducing a simplified rotation gate for updating Q-bits and a decreasing rotation angle approach for determining the magnitude of the rotation angle. The current best known results for problems in case study 1 have been reported in these works. A Memetic Algorithm (MA) and a Genetic Algorithm (GA) using local search combined with Lagrangian relaxation are introduced in [37]. In these algorithms, a local search is integrated as part of the reproductive mechanism. Results show that this approach can yield reasonable schedules at

satisfactory computational times. Although it was used to solve problems in case studies 1 and 5, only for the latter it is competitive. GA solutions to the UC problem have been given in [19] with the addition of the problem specific operators. Problem specific operators are defined within windows, thus they act on building blocks rather than bits. Therefore, once a good building block is found it is preserved through the evolution process. In [2] a GA using a repair mechanism is proposed. To speed-up the algorithm the authors resort to a parallel implementation of such mechanism. Ramp rate limits are always enforced while constructing the solutions, and therefore never violated. However, heuristics are used to enforce load feasibility (enough power is committed) and time feasibility (minimum up/down time). The proposed algorithm has been successfully applied to a real problem with 45 units, see case study 4. Dang and Li [6] decompose the UC problem into two subproblems: in the first one the decisions regarding the units status are made by a GA using a floating-point chromosome representation. Since the encoding and decoding schemes are specific to and based on the load profile type, different problems require different such schemes. In the second subproblem, the production of each on-line unit is determined by LR. In [35] a real coded GA is proposed. A solution is represented by a real number matrix, representing the generation schedule for each unit at each time period. A repair mechanism is used to guarantee that the generation schedule satisfies system and unit constraints. The method was tested by using the most common benchmark problems (case study 1) and a 38-units problem (case study 2), being only competitive for the latter one. A very recent type of evolutionary algorithm, the Imperialist Competition Algorithm (ICA), has been applied to the UC problem in [15]. In the ICA a population consists of a set of countries, all divided between imperialist countries and colonies, based on the imperialists power, which is inversely proportional to its cost function for a minimization problem. Then the colonies move toward their relevant imperialist and the position of the imperialists will be updated if necessary. In the next stage, the imperialistic competition among the empires begins, and through this competition, the weak empires are eliminated. The imperialistic competition will gradually lead to an increase in the power of powerful empires and a decrease in the power of weaker ones, until just one empire remains. The authors tested their methodology on the most commonly used benchmark problems, see case study 1. However, as it can be seen in the results section they only improve upon literature results for the problem instance with 10 units. More details on these methods and other developed applications for the UC problem can be found in the extensive and comprehensive bibliographic surveys published over the years [32, 24, 25, 31].

4. THE PROPOSED METHODOLOGY

In recent years many optimization approaches have been developed, one of the most popular being GAs. Typically, GAs evolve a population of solutions as the result of selection, competition, and recombination. Crossover and mutation are used to maintain a diversity of the evolving population and thus, escape from local optima. Several GAs have been proposed for the UC problem, see e.g. [19, 2, 35, 6, 1],. GAs are a powerful stochastic global search technique as the search is performed by exploiting information sampled from different regions of the solution space [27]. Nevertheless, GAs usually do not perform well in fine-tuning near local optimal solutions because they use minimum a priori knowledge and fail to exploit local information. Local Search algorithms start with an initial solution and try to reach an optimal solution by means of small perturbations to the current solution, that is, the search is done within a pre-specified neighborhood. The inclusion of a Local Search procedure into a GA often leads to substantial improvement since the "local" improvement capabilities of the former are being combined with the "global" nature of the GA. GAs with random keys were first introduced by Bean [4], for solving sequencing problems. A Biased Random Key GA (BRKGA) differs from a random key GA in the way parents are selected for mating and also on the probability of inheriting chromosomes from the best parent. In this work, we propose a BRKGA, which is an improvement of the work presented in [29], based on the framework proposed by Gonçalves and Resende in [14]. In here, we use improved decoding and repair mechanisms. In addition, we have incorporated a local search procedure that, as it can be seen in the results section, has lead to better solutions. Chromosomes are represented as vectors of randomly generated real numbers in the interval $[0,1]$. The vector size N is given by the number of generating units. Each component of the vector corresponds to a priority that is to be assigned to each generation unit. The initial population consists of p vectors of N random keys, which are used by the decoder to generate feasible solutions, details are provided in Section 4.1. Then, each solution is evaluated according to its corresponding total cost. Based on this cost, the population is divided into two subsets: the elite set, consisting of the best solutions, and the non-elite set, consisting of the remaining solutions. Solutions in the elite set are copied onto the next generation, which also consists of two other groups of solutions: solutions generated by crossover and new randomly generated solutions. Regarding the former they are obtained by reproduction between a parent taken from the elite solution set and a parent taken from the remaining solutions. Furthermore, the probability of inheriting alleles from the elite parent is higher than

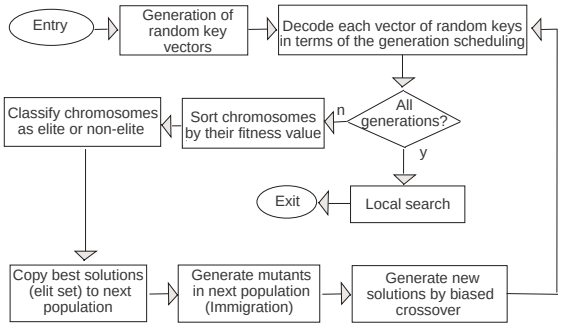


Figure 1: The BRKGA adapted framework.

that of the other parent.

After a sufficiently large number of generations have been evolved we apply a local search procedure to the elite solutions. This leads to obtain better solutions. The BRKGA framework is illustrated in Figure 1, an adaptation from [14].

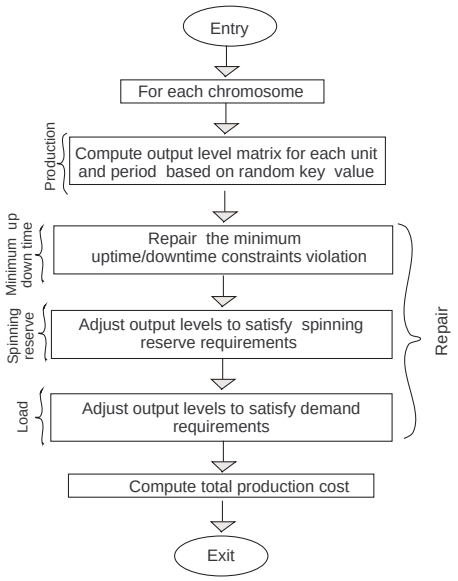


Figure 2: Flow chart of the decoder.

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 unit output range and ramp rate limits for each period is obtained. In this solution, the units are turned online according to their priority, which is given by the associated random key value. Furthermore, unit production is also set by random key value. The production values are chosen such that the ramp rate constraints and the output range constraints are satisfied. Then, these solutions are checked for the remaining constraints and repaired if necessary.

4.1. Decoding Procedure

The decoding procedure proposed here is based on that of [29], however here we do some changes, mainly in the way as the output generation levels are computed. These values are also obtained based on the vector of random keys but now the capacity limits and ramp rate limits satisfaction are ensured during the decoding phase. Therefore, it is not necessary to use the capacity and ramp rate repair mechanisms, the two first repair algorithms used in [29].

Given a vector of numbers in the interval $[0,1]$, say $RK = (r_1, r_2, ..., r_N)$, a rank vector $O = (O_1, O_2, ..., O_N)$ is computed. 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 $j = O_i, i = 1, ..., N$ at time period $t = 1, ..., T$. This amount, which is proportional to the random key value, is guaranteed to be in the range defined by minimum and maximum allowed output limits and ramp rate limits, as follows:

$$y_{t,j} = Y_{t,j}^{min} + r_j . \left(Y_{t,j}^{max} - Y_{t,j}^{min}\right), \tag{9}$$

where $Y_{1,j}^{max} = Ymax_j$, $Y_{1,j}^{min} = Ymin_j$. These limits are defined considering the unit output generation level limits and the ramp rate limits. At the same time that the ramp rate constraints 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 not only on the unit output limits but also on the output level of the current period t. Equation (10) show how these values are obtained.

$$Y_{t,j}^{max} = min\left\{Ymax_j, y_{t-1,j} + \Delta_j^{up}\right\}, \;\; Y_{t,j}^{min} = max\left\{Ymin_j, y_{t-1,j} - \Delta_j^{dn}\right\}. \tag{10}$$

After computing the output generation matrix Y, with the production level of each unit j for each time period t, the generation schedule may not be admissible and therefore, the solution obtained may be infeasible. Hence, the decoding procedure also incorporates a repair mechanism.

The repair mechanism starts by ensuring 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 turned off unit, respectively.

PSfrag replacements

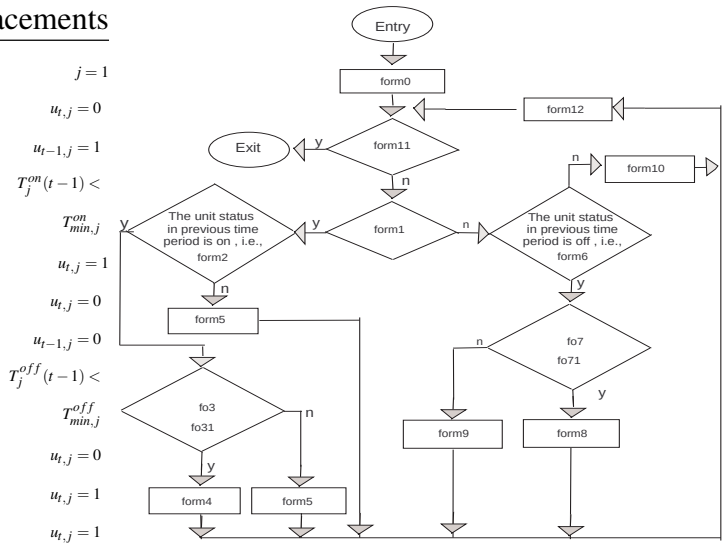


Figure 3: Flowchart of Minimum up down time repair algorithm.

For each period, it may happen that the spinning reserve requirements are not satisfied. If the number of on-line units is not enough, some off-line units must 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. Units are considered for turning off-line in ascending order of priority. At the end of this procedure we have found the U matrix, specifying which units are being operated at each time period, and the Y matrix, which indicates how much each on-line unit is producing. All constraints are satisfied except, may be, the load demand. Nevertheless, the maximum and minimum allowed production limits can be directly inferred from matrix Y. Therefore, we must adjust the total production to satisfy load demand for each time period. Firstly, all on-line units production is set to its minimum allowed value. Next, for each time period, each unit is set to its maximum allowed production, one at the time, until the production

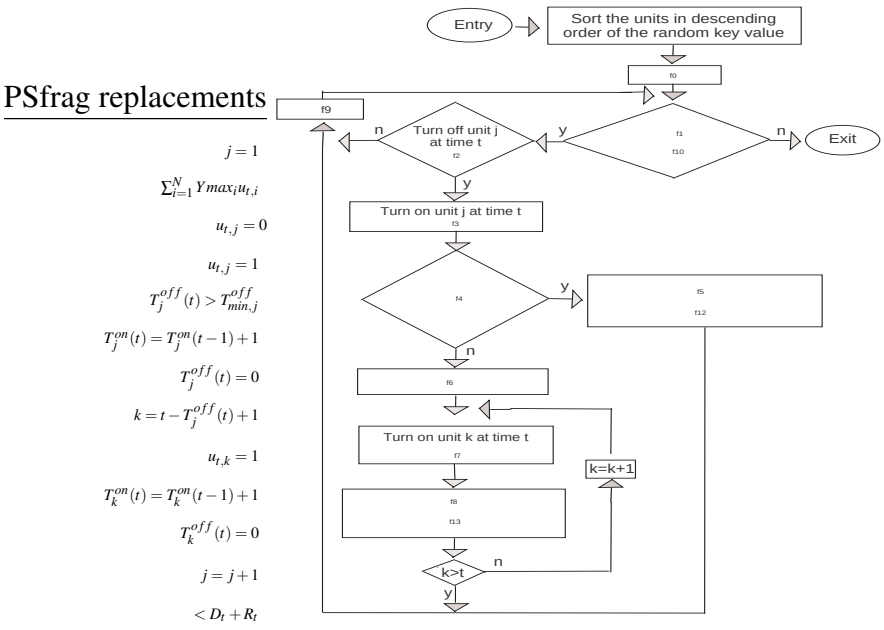


Figure 4: Handling spinning reserve constraint.

reaches the load demand value. In doing so, units are considered in descending order of priority, i.e. random key value. This is repeated no more than N times. It should be noticed that by changing production at time period t the production limits at time period $t+1$ change, and hence these new values, which are obtained as in equation (10), must be satisfied. Once these repairing procedures have been performed, the feasible solution obtained is evaluated through its respective total cost.

4.2. GA Configuration

To obtain a new population of solutions we join in 3 subsets of solutions obtained as follows:

- Copied Solutions: 20% of the best solutions of the population of the current generation (elite set) are copied onto the population of the next generation;
- Mutants: 20% of the solutions of the population of the next generation are obtained by randomly generating new solutions. New Random Keys are generated and used to obtain these solutions;

- Offspring Solutions: 60% of the solutions of the population of the next generation are obtained by biased reproduction, which is achieved by using both a biased parent selection and a biased crossover probability.

As said before, the biased reproduction is accomplished by using both a biased parent selection and a biased crossover. Biased parent selection is performed by randomly choosing one of the parents from the elite set and the other parent from the remaining solutions. This way, elite solutions are given a higher chance of mating, and therefore of passing on their characteristics to future populations. Regarding the biased crossover, we consider a biased coin which is tossed to decide on which parent to take the gene from. Since the coin is biased, the offspring inherits the genes from the elite parent with higher probability (0.7 in our case).

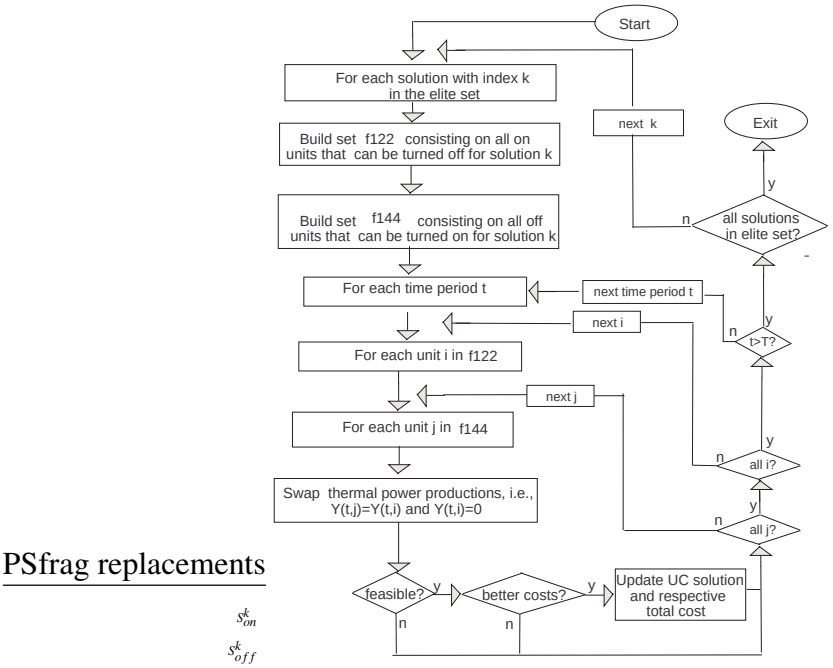
4.3. Local Search

Another improvement to our previous work [29] is the inclusion of a local search procedure. At the end of BRKGA we use a local search procedure to try to improve the solutions in the final elite set. This mechanism, which is illustrated in Figure 5, is a 2-swap procedure, where an on-line generation unit is replaced by an off-line generation unit, if the swap is feasible and leads to a lower cost. Given a solution (in the elite set) we build two sets of generation units. On the one hand, we build a set S_{on} containing the on-line generation units that can be turned off; on the other hand, we build a set S_{off} containing the off-line units that can be turned on.

For each time period, we pick-up a pair of units, one from each of the sets built, and analyze the feasibility of the swap. If the swap is feasible, we compare the total cost of the new solution with that of the current solution. If an improvement can be achieved, the swap is performed resulting in a better solution; otherwise the swap is discarded. In both cases, we move on and try the next swap using the previously built sets, i.e. no update to the sets S_{on} and S_{off} is performed. The 2-swap strategy is repeatedly performed until all swaps have been tried. The procedure is applied to all solutions in the elite set. The contribution of the local search to the global solution quality can be seen in the results provided Section 6.

5. MIXED INTEGER QUADRATIC PROGRAMMING

The UC problem can be casted as a mixed-integer nonlinear program (MINLP). Despite the ever-increasing availability of cheap computing power and the advances in off-the-shelf software for MINLP, solving (UC) by general-purpose soft-



PSfrag replacements

s^k_{on}

s^k_{off}

Figure 5: Flow chart of local search.

ware, even using the most advanced approaches available, is not feasible when the number of units and/or the length of the time horizon becomes large [9].

Here and for comparison purposes we formulate the UC problem as a Mixed Integer Quadratic Programming (MIQP) problem and solve it using the commercial software CPLEX. To do so we simply pass the MIQP formulation given below to CPLEX.

In order to formulate the UC problem as a MIQP model we need to introduce the following auxiliary binary variables:

$l_{t,j}$: binary variable which indicates wether unit j has been started-up or not at time period t (1 if it has been started-up; 0 otherwise);

$h_{t,j}$: binary variable which indicates the cold status of the off-line unit j at time t (1 if the unit is cold; 0 otherwise);

$v_{t,j}$: binary variable which indicates wether unit j has had a cold start-up or not at time period t(1 if it had; 0 otherwise).

The objective function is now rewritten as

$$\text{Min} \sum_{t=1}^{T} \left(\sum_{j=1}^{N} \left\{ a_j \cdot (y_{t,j})^2 + b_j \cdot y_{t,j} + c_j \cdot u_{t,j} + S_{H,j} \cdot l_{t,j} + \left(S_{C,j} - S_{H,j} \right) \cdot v_{t,j} \right\} \right).$$

As before several constraints must be satisfied. The power balance, the spinning reserve, the minimum and maximum production capacity and the ramp rate constraints are express as before, see equations (4) to (7) in Section 2.

The minimum up time constraints are nonlinear and thus are reformulated as given in equation (11).

$$\sum_{k=t}^{t_{max}} u_{k,j} \geq \left(u_{t,j} - u_{t-1,j} \right) \cdot t_s, \text{ for } t \in \{1,...,T\} \text{ and } j \in \{1,2,...,N\}. \qquad (11)$$

where

$$t_s = \begin{cases} min\left\{ T^{on}_{min,j}, T-t+1 \right\}, & \texttt{if } t>1 \texttt{ or } (t=1 \texttt{ and } Instatus(j)<0), \\ max\left\{ 0, T^{on}_{min,j} - Instatus(j) \right\}, & \texttt{if } t=1 \texttt{ and } Instatus(j)>0, \end{cases}$$

$$t_{max} = \begin{cases} min\left\{ t+t_s-1, T \right\}, & \texttt{if } t_s>0, \\ T, & otherwise. \end{cases}$$

The minimum down time constraints are also nonlinear and thus are reformulated as given in equation (12).

$$\sum_{k=t}^{t_{max}} \left(1 - u_{k,j} \right) \geq \left(u_{t,j} - u_{t-1,j} \right) \cdot t_s, \text{ for } t \in \{1,...,T\} \text{ and } j \in \{1,2,...,N\}. \quad (12)$$

where

$$t_s = \begin{cases} min\left\{ T^{off}_{min,j}, T-t+1 \right\}, & \texttt{if } t>1 \texttt{ or } (t=1 \texttt{ and } Instatus(j)>0), \\ max\left\{ 0, T^{off}_{min,j} + Instatus(j) \right\}, & \texttt{if } t=1 \texttt{ and } Instatus(j)<0, \end{cases}$$

$$t_{max} = \begin{cases} min\left\{ t+t_s-1, T \right\}, & \texttt{if } t_s>0, \\ T, & otherwise. \end{cases}$$

Given the newly defined variables, we need to define the following new constrains:

$$l_{t,j} \geq u_{t,j} - u_{t-1,j}, \text{ for } t \in \{1,2,...,T\} \text{ and } j \in \{1,2,...,N\}. \quad (13)$$

$$v_{t,j} \geq l_{t,j} + h_{t,j} - 1, \text{ for } t \in \{1,2,...,T\} \text{ and } j \in \{1,2,...,N\}. \quad (14)$$

$$h_{t,j} \geq 1 - \sum_{k=t_{min}}^{t} \left(u_{k,j}\right), \text{ for } t \in \{1,2,...,T\} \text{ and } j \in \{1,2,...,N\}. \quad (15)$$

where

$$t_{min} = \begin{cases} t - t_l, & \texttt{if } Instatus(j) > 0 \texttt{ or } t > t_l \\ 1, & \texttt{if } Instatus(j) < 0 \texttt{ and } t \leq t_l \texttt{ and } t - t_l - 1 - Instatus(j) \geq 0, \end{cases}$$

with $t_l = T^{off}_{min,j} + T_{c,j} + 1$.

Constraints (13) guarantee that unit j has been started at time t only if it is on at time time t and has been off at time $t-1$. In equation (14) it is assured that the cold start costs are only paid if unit j is cold and has been just started. Finally, constraints (15) state that unit j is cold at time t if and only if it has not been started for at least T^{off} time periods.

CPLEX can be attractive in many situations since in addition to its robustness, it also allows for the incorporation of other constraints [9]. However, since for solving problems with integer variables CPLEX uses a Branch-and-Cut algorithm, it ends up solving a series of continuous subproblems. To these subproblems cuts must be added, on fractional-valued variables of the solution to the subproblems, in order to generate new subproblems with more restrictive bounds on the branching variables. Thus, a single mixed integer problem is decomposed into many subproblems. Therefore, even small sized problems require significant amounts of time and physical memory to be solved. Furthermore, CPLEX cannot cope with more general cost functions, such as, for example, exponential start-up costs, as is the case of the problems in case study 5, first proposed by [36] and [3].

6. NUMERICAL RESULTS

In this section, we report on the results obtained for the BRKGA and for the BRKGA with local search incorporated at the final generation, here and thereafter referred to as IBRKGA. It should be noticed that the parameters are the same for both algorithms. The proposed approaches have been tested on 5 different

benchmark UC case studies. Some of the case studies include several problem instances, while others include only one. Amongst the case studies considered we single out case study 1, since the problems it consists of are the ones that have been consistently considered in the literature and thus solved by many different methods and authors. Due to the stochastic nature of the methods proposed each problem was solved 20 times. Both GAs were implemented in Matlab.

6.1. GA parameters setting

The present state-of-the-art theory on GAs does not provide information on how to configure them. Therefore, the values used in our computational experiments have been taken from the guidelines provided in [14], as well as, from past experience [29]. Computational experiments with different values for the crossover probability, the number of generations, and the population size. These experiments were conducted on the problem with 40 generations units to be scheduled over a 24-hour period given in case study 1.

The biased crossover probability was tested on the range $0.5 \leq P_c \leq 0.9$ with a step size of 0.1. These 5 values were tried for 5 different populations sizes ($NP = n$, $2N$, $3N$, $4N$, and $5N$). Regarding the number of generations, it was set, for test purposes, to a sufficiently large number ($NGers = 20N$), which soon became known to be too large and thus was reduced to $10N$, see Figure 6. To illustrate the algorithm behavior in Table 1 we give the results obtained for varying P_c values with $NP = 2N$ and $NGers = 10N$. The results show no major differences in the algorithm performance. We chose the value 0.7 since to this value corresponds the best performances with lower variability. (Note that a better best solution was found using 0.6.)

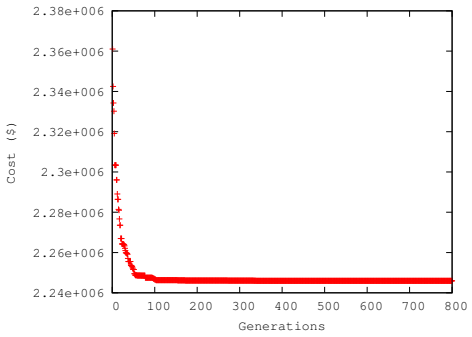


Figure 6: Average cost for the 40-unit system (case study 1) by generation.

Table 1: Average cost for the 40-unit system (case study 1) for different crossover probability values.

P_c	Best	Average	Worst	$\frac{Average-Best}{Best}\%$	$\frac{Worst-Best}{Best}\%$
0.5	2244466	2245409	2246047	0.04	0.07
0.6	2244312	2245388	2247529	0.05	0.14
0.7	2244345	2245350	2245775	0.04	0.06
0.8	2244347	2245432	2246957	0.05	0.12
0.9	2244354	2245476	2246566	0.05	0.10

Regarding the population size NP, as it can be seen from Figure 7, the solution quality is continuously and marginally improved when increasing the population size while the computation time is almost linearly increased. A trade-off analysis between the solution quality and the computation time lead us to choose for NP the value $2N$.

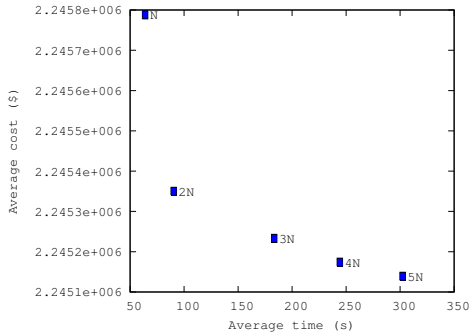


Figure 7: Average cost and computational time for the 40-unit system (case study 1) for different population sizes.

In summary, we have set the number of generations to $10N$, the crossover probability to 0.7, and the population size to $2N$.

In the following sections we compare the results obtained by BRKGA and IBRKGA with the methods reported in the literature that have the best known results. Furthermore, we have used CPLEX (version 12.1) to obtain an optimal solution and thus find out how close our results are to the optimum. Nevertheless, such comparisons are only possible for small sized problems, since CPLEX is unable to solve larger problems due to the huge memory requirements. In addition, CPLEX cannot handle problems where the start-up costs are given by an exponential function of the number of hours that unit has been down, as is the case of

Table 2: Comparison of the best results obtained by the BRKGA and the IBRKGA with the best ones reported in literature, for problems of case study 1.

	[39]	[17]	[20]	[15]			CPLEX	IBRKGA		
Size	IPSO	IQEA	QEA	ICA	BRKGA	IBRKGA	MIQP	Rank	Gap(opt)	Gap(best)
10	563954	563977	***563938***	***563938***	564248	**563938**	563938	1st	0	0
20	1125279	1123890	***1123607***	1124274	1124664	1123955	1123297	3rd	0.06	0.03
40	2248163	*2245151*	2245557	2247078	2244492	**2244345**	2242634	1st	0.08*	-0.04
60	3370979	*3365003*	3366676	3371722	3365026	**3363804**	–	1st	–	-0.04
80	4495032	*4486963*	4488470	4497919	4486833	**4485197**	–	1st	–	-0.04
100	5619284	*5606022*	5609550	5617913	5607288	**5605933**	–	1st	–	-0.002

* Recall that this is the best known solution, although it may not be an optimal solution.

the problems in case study 5 used in this work.

6.2. Case study 1

The BRKGA proposed was tested on a set of often used benchmark problems, involving systems with 10 up to 100 generation units and considering, in each case, a horizon of 24 hours. The 10 generation unit system problem, the base case, was originally proposed by [3]. Subsequential, problem instances involving 20, 40, 60, 80 and 100 units are obtained by replicating the base case system and the load demands are adjusted in proportion to the system size. In all cases the spinning reserve is kept at 10% of the hourly demand. Details of how these benchmark problems were constructed and on the system and demand data can be found in [19].

For the problems in this case study, CPLEX was able to find an optimal solution to systems involving 10 and 20 units. For problems with 40 units, we report on the best solution found by CPLEX before it crashed due to the excessive memory requirements. However, although the solution is not optimal, it is the best solution found so far. In tables 2 to 4 we compare the results obtained (best, average, and worst) with the best former results (in italic) obtained amongst the many publish methods. In best current solution, excluding the CPLEX one, is given in bold, for each of the problems. In the last column, we report on the gap between the IBRKGA solution and the previously best known solution. It should be noticed that whenever the IBRKGA produces a solution which is better than the best currently known solution the gap is negative. In Table 2 we also report on the optimality gap for the smaller problem instances, since for these we have the optimal solution value (provided by CPLEX).

As it can be seen in Table 2, for all problem instances, except one, our best results improve upon the best previously known results. Moreover, for the problem

Table 3: Comparison of the best results obtained by the BRKGA and the IBRKGA with the best average ones reported in literature, for problems of case study 1.

	[17]	[20]			IBRKGA	
Size	IQEA	QEA	BRKGA	IBRKGA	Rank	Gap(%)
10	563977	***563969***	564445	564062	2nd	0.02
20	*1124320*	1124689	1124846	**1124213**	1st	-0.01
40	*2246026*	2246728	2245820	**2245350**	1st	-0.03
60	*3365667*	3368220	3366053	**3365201**	1st	-0.02
80	*4487985*	4490128	4488303	**4487620**	1st	-0.01
100	*5607561*	5611797	5607902	**5607024**	1st	-0.01

Table 4: Comparison of the best results obtained by the BRKGA and the IBRKGA with the best worst ones reported in literature, for problems of case study 1.

	[39]	[17]	[20]			IBRKGA	
Size	IPSO	IQEA	QEA	BRKGA	IBRKGA	Rank	Gap(%)
10	564579	***563977***	564672	565689	564737	4th	0.135
20	1127643	***1124504***	1125715	1126273	1125048	2nd	0.048
40	2252117	*2246701*	2248296	2246797	**2245775**	1st	-0.04
60	3379125	***3366223***	3372007	3367777	3366773	2nd	0.016
80	4508943	*4489286*	4492839	4489663	**4488962**	1st	-0.01
100	5633021	***5608525***	5613220	5609537	5608559	2nd	0.001

instances for which an optimal solution has been found by CPLEX it can be seen that the IBRKGA has been able to find an optimal solution in one case, while in the other case the solution found is within 0.06% of optimality. By comparing the IBRKGA with the BRKGA, which already improves upon some of the previously known best solutions, we can see that the local search is always effective since the IBRKGA is always better than the BRKGA. And the improvement ranges from 0.007% to 0.063%. Although these values are small their impact may be very relevant given that they refer to a multi-million dollar industry.

Regarding the average results we have also improved on all but one of the problem instances solved, when compared with the best previously known results, see Table 3. In Table 4 similar results are reported for the worst solutions. In this case, we still have improved upon previous results, but only for two problem instances.

The results reported in these tables also show that the local search incorporation is effective, since the IBRKGA improves upon the BRKGA. On average, the average and the worst solutions are improved by 0.034% and by 0.064%, respectively.

Table 5: Analysis of the variability of the solution quality for problems of case study 1.

Size	$\frac{Average-Best}{Best}$%			$\frac{Worst-Best}{Best}$%				St. deviation(%)		
		[17]	[20]		[39]	[17]	[20]		[17]	[20]
	IBRKGA	IQEA	QEA	IBRKGA	IPSO	IQEA	QEA	IBRKGA	IQEA	QEA
10	0.02	0.0	0.005	0.14	0.11	0.0	0.13	0.03	0.0	0.02
20	0.02	0.04	0.09	0.1	0.21	0.05	0.19	0.03	0.01	0.06
40	0.04	0.04	0.05	0.06	0.18	0.07	0.12	0.02	0.02	0.02
60	0.04	0.02	0.05	0.09	0.24	0.04	0.16	0.02	0.01	0.03
80	0.05	0.02	0.04	0.08	0.31	0.05	0.1	0.02	0.01	0.02
100	0.02	0.03	0.04	0.05	0.24	0.04	0.07	0.01	0.01	0.02

Another important feature of the proposed algorithm is that, as it can be seen in Table 5, the variability of the results is very small. The difference between the worst and best solutions found for each problem is always below 0.14%, while if the best and the average solutions are compared this difference is never larger than 0.05%. This allows for inferring the robustness of the solution since the gaps between the best and the worst solutions are very small. This is very important since the industry is reluctant to use methods with high variability as this may lead to poor solutions being used. When compared to the robustness of the alternative methods, it can be seen that it is better than that of the IPSO and QEA and almost the same as that of the IQEA.

Regarding the computational time, no exact comparisons may be done since, on the one hand, the values are obtained on different hardware, and on the other hand, the IBRKGA reported time is real time and not CPU time and thus it is not directly comparable with others reported in the literature. Our computational experiments were performed on a Xeon X5450, 3.0 GHz and 4.0 GB RAM. This is a shared machine and therefore several jobs are usually running in parallel. Nevertheless, in Table 6 we report on our computational time requirements as well as on the ones of the works used for comparison purposes. It should be noticed that the results reported for [39, 17] may not be accurate, since the authors only provide them in a graphical form. These results are also provided graphically, on the left hand side of Figure 8. As it can be seen, the IPSO has computational time requirements much larger than the other methods. On the contrary, the QEA has a much better performance. The other three methods have a similar behavior in what concerns computational requirements. Therefore, the IBRKGA has an intermediate performance, regarding computational time, which does not seem to be a big price to pay for increased solution quality. Recall that, as seen in Table 2,

Table 6: Analysis of the execution time, for problems in case study 1.

Size	IPSO [39]	IQEA [17]	QEA [20]	ICA [15]	BRKGA	IBRKGA	CPLEX
10	142	15	19	48	2	2	45
20	357	42	28	63	13	14	401
40	1100	132	43	151	87	90	1489
60	2020	273	54	366	276	301	–
80	3600	453	66	994	631	712	–
100	5800	710	80	1376	1259	1503	–

the BRKGA provides the best solution for all but one of the problems analyzed in this case study.

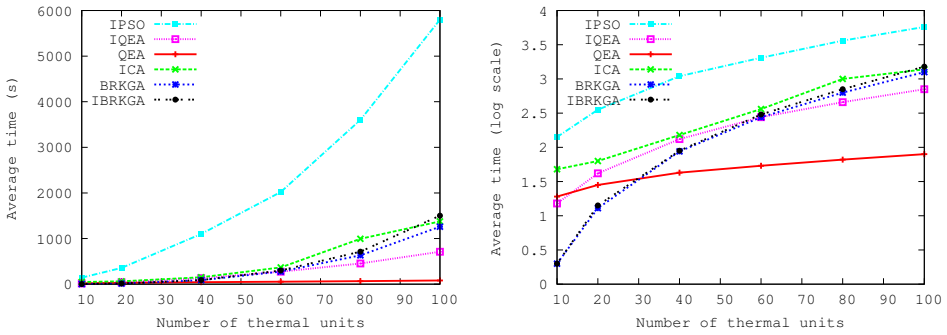


Figure 8: Computational time requirements, in seconds and log scale, for the methods being compared, for problems in case study 1.

When we analyze the computational time in a logarithmic scale, see the graph on the right hand side of Figure 8, a favorable conclusion regarding our algorithms can be drawn. The growth of all the other algorithms is closer to a line in the log scale, meaning that the time increase with problem size is closer to an exponential growth. In our algorithms, it is notorious that the growth is more concave in the log scale, meaning that the time increase is subexponential.

6.3. Case study 2

Case study 2 consists of a single real problem instance, the Taipower system, which comprises the scheduling of 38 units for a time horizon of 24 hours. This problem was first proposed by [16]. The start up costs are considered constants, while shut down costs are disregarded. The spinning reserve was set to 11% of hourly load and the ramp rate constraints are also taken into consideration. The

characteristics of the thermal units, the load demand, and the specific conditions of the problem are given in [16]. This specific problem has not been considered by many authors doing research of the UC problem. Thus, our approach is compared with the four approaches proposed in [16] which are based on dynamic programming (DP), Lagrangian Relaxation (LR), Simulated Annealing (SA), and Constraint Logic Programming (CLP) and also with a GA (MRCGA) recently proposed in [35]. In addition, we also compare our solutions to the solution we have obtained by using CPLEX. This solution may not be optimal since CPLEX has not ran to the end due to the excessive memory requirement. However, we report on the best solution found, before it crashed, and on the time it took to find such a solution for the first time.

Table 7: Comparison of the results obtained by the BRKGA and the IBRKGA with the best ones reported in literature, for problems of case study 2.

Size	DP [16]	LR[16]	CLP[16]	MRCGA [35]	BRKGA	IBRKGA	CPLEX
Best	215.2	214.5	213.8	*206.7*	206.0	**205.3**	203.6
Average	–	–	–	*207.4*	206.5	**206.1**	–
Worst	–	–	–	*208.0*	207.1	**206.7**	–
Gap(%)	5.7	5.4	5.0	1.5	1.2	0.83	–
St.deviation(%)	–	–	–	–	0.19	0.22	–
Av.Time(s)	199	29	17	45.6	84.7	102.6	1963.9

Both the BRKGA ad the IBRKGA improve on the best known solutions, for all cases (best,average, and worst), due to the GA proposed in [35]. Again the local search has proved to be effective since in all cases the IBRKGA obtains better solutions than the BRKGA. The computational times are not a concern since the method that takes longer (the DP proposed in [16]) takes just over 3 minutes.

6.4. Case study 3

Case study 3 also consists of a single real problem. This problem is a 26-generator system which has to be scheduled for a 24-hour period. The system and demand data can be found in [38], as well as, the conditions used in the computational experiments. We compare the solution quality obtained by the BRKGA and by the IBRKGA with that a fuzzy mixed integer Linear Programming proposed in [38]. For this problem we were able to find an optimal solution by using CPLEX. As it can be seen in Table 8 both the BRKGA and the IBRKGA improve on the previously best known results. Again the use of the local search allowed for obtaining an improved solution. Furthermore, the best solution obtained by

the IBRKGA is very close to an optimal solution. Thus, CPLEX is not a good alternative when compared with the IBRKGA since, the latter obtains a solution within 0.28% of optimality and is about 21 times faster.

Table 8: Comparison of the results obtained by the BRKGA and the IBRKGA with the best ones reported in literature, for problems of case study 3.

Size	FMILP [38]	BRKGA	IBRKGA	CPLEX
Best	*722388*	721383	**721297**	719314
Average	–	721410	**721322**	–
Worst	–	721488	**721401**	–
Gap (%)	0.42	0.29	0.28	–
St.deviation(%)	–	0.01	0.01	–
Av.Time(s)	25.5	24.3	29.6	642

6.5. Case study 4

The problem addressed in this case study comprises 45 units over a planning horizon of 24 hours. The system data and the load demand can be found in [2]. The spinning reserve is set to 10% of the load demand at every hour. The start-up costs are assumed constant. Table 9 shows the best solutions know so far, obtained in [2] from three versions of a GA: global parallelization (GP), which uses a parallel implementation of the repair algorithm, coarse-grained parallel genetic algorithm (CGPGA), which evolves several populations independently, one on each processor, and hybrid parallel genetic algorithm (HPGA), which combines both previously parallelizations. In addition, our best, average and worst solutions, for both the BRKGA and the IBRKGA, are reported. The methods here proposed improve on the best known solution by 0.22%. For this problem the local search was not effective, since the best, average , and worst solutions of the IBRKGA have the same value of those of the BRKGA. It should be notice that CPLEX was unable to provide any solution for this problem due to its size.

Regarding the computational time, although our approaches have not been implemented in parallel, they are faster than the approach producing the best former results.

6.6. Case study 5

Case study 5 consists of a two different problems both considering exponential start-up costs. This type of cost are more realistic and although several authors mention this, they all end up using constant cost or otherwise approximating them by a piecewise linear function.

Table 9: Comparison of the results obtained by the BRKGA and the IBRKGA with the best ones reported in literature, for problems of case study 4.

Size	GP [2]	CGPGA[2]	HPGA [2]	BRKGA	IBRKGA
Best	1034472374	1032472928	*1032415327*	**1030145017**	**1030145017**
Average	–	–	–	**1030722315**	**1030722315**
Worst	–	–	–	**1034934856**	**1034934856**
Gap (%)	0.42	0.23	0.22	0	–
St.deviation(%)	–			0.14	0.14
Av.Time(s)	80.6	847.1	658.4	115.6	147.3

Both problems in this case study involve the scheduling of 10 units over a 24-hour time horizon. The first problem has been proposed by [36], where all problem data is given.

The start-up costs are computed as:

$$S_{t,j} = b_0 . \left(1 - b_1 . e^{-b_2 t}\right). \tag{16}$$

This problem has been addressed in [37], where an optimal solution has been found by using dynamic programming. Furthermore, the authors also propose approximate methods to address this problem: a Lagrangian Relaxation (LR), a genetic algorithm (GA), a memetic algorithm (MA), and a method combining both the LR and MA (LRMA).

In Table 10 we report the result published in [37], as well as, the results obtained by our approaches. As it can be seen, we are able to obtain a good solution (with a 0.51% optimality gap), which is better than that of the GA, the MA, and the LRMA proposed in [37]. However, the LR was able to find a better solution. Regarding computational time, our methodologies are much better, being up to 53 times faster. For this problem, it happens again that the local search does not help in finding a better solution.

Table 10: Comparison of the results obtained by the BRKGA and the IBRKGA with the best ones reported in literature, for the first problem of case study 5 problem 1.

Size	DP[37]	LR [37]	GA [37]	MA[37]	LRMA [37]	BRKGA	IBRKGA
Best	59478	***59485***	59882	59788	59892	59779	59779
Average	–	***59486***	60364	60271	59936	59836	59834
Worst	–	***59491***	60977	60838	60100	60102	60091
Gap (%)	–	0.01	0.68	0.52	0.7	0.51	0.51
St.deviation(%)	–	0.004	0.74	0.65	0.123	0.11	0.11
Av.Time(s)	207	55	209	161	128	3.9	4.7

The second problem in this case study is also a problem involving 10 units that must be scheduled over a 24-hour period. This problem has been proposed by Bard and the problem data can be found in [3].

The start-up costs are given as follows:

$$S_{t,j} = b_0 . \left(1 - e^{-\frac{max(0,-T_j^{off}(t))}{b_2}} \right) + b_1 . \tag{17}$$

More recently other authors have addressed this problem. The results they have obtained are compared to the ones we have obtained, see Table 11.

The more recently proposed heuristics, the DP and MA in [37], the FPGA in [6], and ours, were not able to improve on the best known result found in [3] by using a LR approach. Regarding the quality of the average and of the worst solution, the IBRKGA is the method that provides the best results. It should be noticed that the BRKGA already presents better average and worst results than the other heuristics. Therefore, the BRKGA and the IBRKGA methods present solutions with the lowest variability. Moreover, the BRKGA and IBRKGA average execution times are shorter than that of the other methods, the IBRKGA being up to 43 times faster than the DP heuristic. For this problem the local search is effective, since the IBRKGA solution quality is better for all solution types.

Table 11: Comparison of the results obtained by the BRKGA and the IBRKGA with the best ones reported in literature, for the second problem of case study 5 problem 2.

Size	DP [37]	LR[3]	MA[37]	FPGA [6]	BRKGA	IBRKGA
Best	540904	***540895***	541108	541182	542068	541918
Average	–	–	545591	542911	542508	**542372**
Worst	–	–	549290	545572	543377	**543301**
Gap (%)	0.002	–	0.04	0.05	0.21	0.19
St.deviation(%)	–	–	0.61	0.27	0.1	0.11
Av.Time(s)	255	59	101	–	5.9	7.3

7. CONCLUSIONS

Biased Random Key GAs have been developed for and applied to several combinatorial optimization problems with interesting results. Given this empirical evidence, see [14], we previously proposed such an algorithm for the unit commitment problem [29]. The results obtained suggested that such an approach would worth while of further investigation. Therefore, in this paper, we propose a Biased

Random Key Genetic Algorithm with Local Search to address the unit commitment problem. In addition, we have improved the decoding and repair procedures used within the GA.

The new algorithm has been tested on a set of UC benchmark problems commonly used and other UC problems found in the literature. The results reported here, show that the proposed method outperforms the current state-of-the-art methods available. For all problem instances, but two, we have been able to find better results then the best results found so far. In addition, these better solutions have been found with computational time requirements, typically, smaller or of the same magnitude than that alternative methods. Furthermore, the results show a further very important feature, lower variability. It should be noticed that the difference between the best and the worst solutions is always below 0.14%, while the difference between the best and the average solutions is always below 0.05%, for the most commonly used problems (case study 1). This is very important since the methods to be used in industrial applications are required to be robust, since otherwise they may lead to poor solutions being used.

[1] K. Abookazemi, M. Mustafa, and H. Ahmad. Structured Genetic Algorithm Technique for Unit Commitment Problem. *Int. J. of Recent Trends in Engineering*, 1(3):135–139, 2009.

[2] J. Arroyo and A. Conejo. A parallel repair genetic algorithm to solve the unit commitment problem. *IEEE Trans. on Power Systems*, 17:1216–1224, 2002.

[3] J. Bard. Short-term scheduling of thermal electric generators using Lagragian Relaxation. *Operation Research*, 36(5):756–766, 1988.

[4] J. Bean. Genetic Algorithms and Random Keys for Sequencing and Optimization. *Operations Research Society of America J. on Computing*, 6(2):154–160, 1994.

[5] A. I. Cohen and M. Yoshimura. A Branch-and-Bound Algorithm for Unit Commitment. *IEEE Trans. on Power Apparatus and Systems*, 102(2):444–451, 1983.

[6] C. Dang and M. Li. A floating-point genetic algorithm for solving the unit commitment problem. *European J. Operational Research*, 181(4):1370–1395, 2007.

[7] W. Fan, Y. Liao, J. Lee, and Y. Kim. Evaluation of Two Lagrangian Dual Optimization Algorithms for Large-Scale Unit Commitment Problems. *J. of Electrical Engineering & Technology*, 7(1):17–22, 2012.

[8] A. Frangioni and C. Gentile. Solving nonlinear single-unit commitment problems with ramping constraints. *Operations Research*, 54(4):767–775, 2006.

[9] A. Frangioni, C. Gentile, and F. Lacalandra. Solving unit commitment problems with general ramp constraints. *Electrical Power and Energy Systems*, 30(5):316–326, 2008.

[10] A. Frangioni, C. Gentile, and F. Lacalandra. Tighter Approximated MILP Formu-

lations for Unit Commitment Problems. *IEEE Trans. on Power Systems*, 24(1):105–113, 2009.

[11] J. Gonçalves. A hybrid genetic algorithm-heuristic for two-dimensional orthogonal packing problem. *European J. Operational Research*, 183(3):1212–1229, 2007.

[12] J. Gonçalves, J. M. Mendes, and M. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European J. Operational Research*, 167(1):77–95, 2005.

[13] J. Gonçalves, J. M. Mendes, and M. Resende. A genetic algorithm for the resource constrained multi-project scheduling problem. *European J. Operational Research*, 189(3):1171–1190, 2008.

[14] J. Gonçalves and M. Resende. Biased random-key genetic algorithms for combinatorial optimization. *J. of Heuristics*, 17(5):487–525, 2010.

[15] M. Hadji and B. Vahidi. A Solution to the Unit Commitment Problem Using Imperialistic Competition Algorithm . *IEEE Trans. on Power Systems*, 27(1):117–124, 2012.

[16] K. Huang, H. Yang, and C. Yang. A new thermal unit commitment approach using constraint logic programming. *IEEE Trans. Power Systems*, 13(3), 1998.

[17] Y. Jeong, J. Park, J. Shin, and K. Lee. A thermal unit commitment approach using an improved quantum evolutionary algorithm. *Electric Power Components and Systems*, 37(7):770–786, 2009.

[18] K. Juste, H. Kita, E. Tanaka, and J. Hasegawa. An evolutionary programming solution to the unit commitment problem. *IEEE Trans. on Power Systems*, 14(4):1452–1459, 1999.

[19] S. Kazarlis, A. Bakirtzis, and V. Petridis. A Genetic Algorithm Solution to the Unit Commitment Problem. *IEEE Trans. on Power Systems*, 11:83–92, 1996.

[20] T. Lau, C. Chung, K. Wong, T. Chung, and S. Ho. Quantum-Inspired Evolutionary Algorithm Approach for Unit Commitment. *IEEE Trans. on Power Systems*, 24(3):1503–1512, 2009.

[21] G. Lauer, N. Sandell, D. Bertsekas, and P. T.A. Solution of large scale optimal unit commitment problems. *IEEE Trans. on Power Apparatus and Systems*, PAS-101(1):79–96, 1982.

[22] Z. Michalewicz and C. Janikow. Handling Constraints in Genetic Algorithms. In *In Belew, R.K., Booker, L.B., eds.: Proceedings of the Fourth Int. Conference on Genetic Algorithms (ICGA-91), San Mateo, California, University of California, San Diego*, pages 151–157. Morgan Kaufmann Publishers, 1991.

[23] J. Muckstadt and S. Koenig. An application of Lagrangian relaxation to scheduling in power-generation systems . *Operations Research*, 25(3):387–403, 1977.

[24] N. Padhy. Unit commitment using hybrid models: a comparative study for dynamic programming, expert system, fuzzy system and genetic algorithms. *Int. J. of Electrical Power & Energy Systems*, 23(8):827–836, 2000.

[25] N. Padhy. Unit Commitment-A Bibliographical Survey. *IEEE Trans. on Power Systems*, 19(2):1196–1205, 2004.

[26] S. Rebennack, P. Pardalos, M. V. Pereira, and N. Iliadis. *Handbook of Power Systems I*. Energy Systems, Springer, 2010.

[27] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. chapter Genetic Algorithms, Blackwell Scientific Publications, 1993.

[28] A. Rong, H. Hakonen, and R. Lahdelma. A variant of the dynamic programming algorithm for unit commitment of combined heat and power systems. *European J. Operational Research*, 190:741–755, 2008.

[29] L. Roque, D. B. M. M. Fontes, and F. A. C. C. Fontes. A Biased Random Key Genetic Algorithm Approach for Unit Commitment Problem. *Lecture Notes in Computer Science*, 6630(1):327–339, 2011.

[30] S. S. Patra, S. Goswami, and B. Goswami. Fuzzy and simulated annealing based dynamic programming for the unit commitment problem. *Expert Systems with Applications*, 36(3):5081–5086, 2009.

[31] S. Salam. Unit commitment solution methods. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 26, pages 600–605, 2007.

[32] S. Sen and D. Kothari. Optimal thermal generating unit commitment: a review. *Electrical Power and Energy Systems*, 20:443–451, 1998.

[33] L. Snyder and M. S. Daskin. A random-key genetic algorithm for the generalized traveling salesman problem. *European J. Operational Research*, 174(1):38–53, 2006.

[34] K. Sourirajan, L. Ozsen, and R. Uzsoy. A genetic algorithm for a single product network design model with lead time and safety stock considerations. *European J. Operational Research*, 197(2):38–53, 2009.

[35] L. Sun, Y. Zhang, and C. Jiang. A matrix real-coded genetic algorithm to the unit commitment problem. *Electric Power Systems Research*, 76:716–728, 2006.

[36] A. Turgeon. Optimal scheduling of thermal generating units. *IEEE Trans. on Automatic Control*, 23:1000–1005, 1978.

[37] J. Valenzuela and A. Smith. A seeded memetic algorithm for large unit commitment problems. *J. of Heuristics*, 8(2):173–195, 2002.

[38] B. Venkatesh, T. Jamtsho, and H. Gooi. Unit commitment - a fuzzy mixed integer linear programming solution. *IET Generation, Transmission and Distribution*, 1(5):836–846, 2007.

[39] B. Zhao, C. Guo, B. Bai, and Y. Cao. An improved particle swarm optimization algorithm for unit commitment. *Int. J. of Electrical Power & Energy Systems*, 28(7):482–490, 2006.