Journal of Combinatorial Optimization. manuscript No.
(will be inserted by the editor)

A Hybrid Biased Random Key Genetic Algorithm Approach for the Unit Commitment Problem

L.A.C. Roque · D.B.M.M. Fontes · F.A.C.C. Fontes

Received: September 16, 2013/ Accepted:

Abstract This work proposes a hybrid genetic algorithm to address the Unit Commitment (UC) problem. In the UC problem, the goal is to schedule a subset of a given group of electrical power generating 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 operational constraints.

The algorithm developed is a Hybrid Biased Random Key Genetic Algorithm (Hybrid BRKGA). The biased random key technique was chosen due to its reported good performance in several combinatorial optimization problems. In the algorithm, solutions are encoded using random keys, which are represented as vectors of real numbers in the interval $[0, 1]$. The proposed GA 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. The BRKGA is hybridized with local search in order to intensify the search close to good solutions.

Tests have been performed on benchmark large-scale power systems with up to 100 units for a 24-hour period. The results obtained have demonstrated that the proposed methodology is an effective and efficient tool for finding solutions to large-scale UC problems. Furthermore, from the comparisons made it is possible to concluded that the results achieved improve the solutions obtained by the state-of-the-art methodologies reported.

DEMA, Instituto Superior de Engenharia do Porto,
4200-072 Porto, Portugal
Tel.: +351 22 83 40 500
E-mail: lar@isep.ipp.pt

LIAAD-INESC-TEC, Faculdade de Economia, Universidade do Porto,
4200-464 Porto, Portugal
Tel: +351 225 571 240
E-mail: fontes@fep.up.pt

ISR-Porto, Faculdade de Engenharia, Universidade do Porto,
4200-465 Porto, Portugal
Tel: +351 22 508 1811
E-mail: faf@fe.up.pt

Keywords Unit Commitment · Genetic Algorithms · Hybrid Metaheuristics · Electrical Power Generation.

1 Introduction

Power systems are one of the most important infrastructures in a country since the commodity involved is essential to everyday life, its availability and price are critical to many companies, and it requires continuous balancing (Rebennack et al, 2010a,b). The study and operation of these systems involves solving many different optimization problems (Kallrath et al, 2009). Amongst these problems, the Unit Commitment (UC) problem stands out as it plays a key role in planning and operating power systems. Optimal scheduling of the generating units not only has the potential of saving millions of dollars, but also of maintaining system reliability by keeping a proper spinning reserve (Zheng et al, 2012). The UC problem is an optimization problem where the goal is to determine the on/off status of the generating units at minimum operating costs. In addition, the production of the committed units, which also needs 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 generating units. Due to its combinatorial nature, multi-period characteristics and nonlinearities, this problem is highly demanding computationally 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. Other versions of the UC problem have also been studied, see for example Zheng et al (2012); Wang et al (2012); Jiang et al (2012); Schneider et al (2013).

In the past, several traditional heuristic approaches based on exact methods have been proposed, such as Dynamic Programming, Branch and Bound, Lagrangian Relaxation and Mixed-Integer Programming, see for example Muckstadt and Koenig (1977); Cohen and Yoshimura (1983); Huang et al (1998); Rong et al (2008); Frangioni et al (2008, 2009); Patra et al (2009). Most of the recently developed methods are metaheuristics, evolutionary algorithms, and hybrids of the them, see for example Zhao et al (2006); Sun et al (2006); Dang and Li (2007); Jeong et al (2009); Lau et al (2009); Roque et al (2011); Hadji and Vahidi (2012). In general, these latter types have led to better results than the ones obtained with the traditional heuristics.

This paper proposes a Hybrid Biased Random Key Genetic Algorithm (HBRKGA) to address the UC problem. The HBRKGA proposed here is based on the framework provided by Gonçalves and Resende (2010), which has been used effectively and efficiently in other important applications (Fontes and Gonçalves, 2007; Gonçalves et al, 2008; Sourirajan et al, 2009; Gonçalves and Resende, 2011; Fontes and Gonçalves, 2012; Kotsireas et al, 2012). The BRKGAs are a variation of the random key genetic algorithms, first introduced by Bean (1994). A Biased Random Key GA differs from a random key GA in the way parents are selected for mating and in the probability of inheriting chromosomes from the best parent. In this HBRKGA, repair mecha-

nisms are also included and therefore all the individuals considered for evaluation are feasible. The HBRKGA is capable of finding better solutions than the best currently known ones for most of the benchmark problems solved. Furthermore, the computational time requirements are modest and similar to those of other recent approaches.

The UC problem is a well-researched combinatorial optimization problem and has been addressed by a large variety of methods. Nevertheless, the approach proposed here has managed to improve current state-of-the-art methodologies. These facts led us to conclude that the proposed approach can be promising to address combinatorial optimization problems in other application areas.

The remainder of this article is organized as follows. Section 2 describes the UC problem and provides its mathematical formulation. Section 3 describes previous methodologies addressing the UC problem is. The solution approach proposed to address the UC problem is explained in Section 4. Then, Section 5 tests the effectiveness and efficiency of the approach proposed here in benchmark systems with up to 100 units for a 24-hour period. In addition, the results obtained are compared to those of the current state of the art approaches reported in the literature. Due to recent advances in MIQP commercial solvers, such as CPLEX, it is possible to solve UC problems optimally, at least for smaller problems. This approach has also been implemented and the results obtained for small size instances are used for comparison purposes. Finally, Section 6 presents some conclusions.

2 UC Problem Formulation

In the Unit Commitment problem the optimal turn-on and turn-off schedules need to be determined over a given time horizon for a group of power generating units under certain operational constraints. In addition, the output levels must be decided for each on-line unit at each time period. The notation is provided prior to the mathematical formulation.

Indexes:	**t**: Time period index; **j**: Generating unit index;
Decision Variables:	$\mathbf{y_{t,j}}$: 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:	**T**: Number of time periods (hours) of the scheduling time horizon; **N**: Number of generating 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 output level decrease/increase allowed in consecutive periods of 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; other-

wise they are set to zero. The real valued variables $y_{t,j}$ 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 minimizing the total operating costs over the scheduling horizon.

2.1 Objective Function

The objective function contains three cost components: generation costs, start-up costs, and shut-down costs. The generation costs, also known as fuel costs, are conventionally given by the following quadratic cost function.

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

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

The start-up costs, which depend on the number of time periods 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. The shut-down costs for each unit S_{dj}, whenever considered in the literature, are not time dependent.

Therefore, the cost incurred with an optimal scheduling is obtained by minimizing the total costs for the entire planning period,

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

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.

1) Load Requirement 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 capacity 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 constraints includes limits on the unit output range, on the maximum output variation allowed for each unit (ramp rate constraints), and on the minimum number of consecutive time periods that the unit must be 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 for 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 switched on or switched 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

This section starts by describing several traditional heuristic approaches based on exact methods that have been reported in the 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.

3.1 Approaches based on exact methods

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 and memory requirements. Due to its enumerative nature, dynamic programming suffers from a long processing time that expands exponentially with the size of the problem. 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 (Sen and Kothari, 1998). More recently, other approximate methods based on DP have been proposed for the UC problem and its variants. For instance Rong et al (2008) have proposed a DP algorithm based on a linear relaxation of the on/off status of the units and on the sequential commitment of the units, one by one, for the UC in multi-period combined heat and power production planning under the deregulated power market. In (Patra et al, 2009), 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 by using heuristics, such as priority ordering of the units, unit grouping, fast economic dispatch based on priority ordering, and avoidance of repeated economic dispatch.

Not many works on the UC problem used Branch-and-Bound (BB). In earlier research (Lauer et al, 1982; Cohen and Yoshimura, 1983), 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, Cohen and Yoshimura (1983) consider that the fuel consumption is given by a linear cost function, which constitutes another major drawback. In (Huang et al, 1998) a two-phase procedure is proposed. In the first phase, using 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.

Lagrangian Relaxation (LR) is capable of solving large scale UC problems swiftly 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 joining the coupling constraints and the cost function using Lagrange multipliers. The resulting relaxed problem minimizes the so-called Lagrangian subjected to the unit constraints. LR was first applied to solve the UC problem without considering ramp constraints (Muckstadt and Koenig, 1977). Bard (1988) uses LR to divide the model into separate subproblems, one for each unit. The author tests the method on a 10-unit system with exponential start-up costs (see case study 5). Recently, in (Frangioni et al, 2008) an effective Lagrangian relaxation approach has been proposed for the UC problem. This approach relies on an exact algorithm for solving the single-unit commitment problem proposed in (Frangioni and Gentile, 2006). More recently, in (Fan et al, 2012) two lagrangian relaxation methods were 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 problem instances. Therefore, no comparisons with alternative methods were possible. From the tests performed, it was possible to conclude that the subgradient method yields better results.

Optimal solutions can be found by solving the MIQP model, but the computational time requirements are enormous and they usually increase exponentially with the problem size, even with the availability of efficient software packages (such as CPLEX and LINDO), as will be seen in the results section. Some authors have tried to improve the performance of 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 for example (Carrion and Arroyo, 2006; Frangioni and Gentile, 2006; Frangioni et al, 2008, 2009; Ostrowski et al, 2012; Viana and Pedroso, 2013). Other authors also relax the time dependent startup cost by considering

a stairwise linear cost function, see for example (Carrion and Arroyo, 2006). Then the linear approximation models are usually solved by the CPLEX. The execution of the CPLEX is stopped when the solutions are within 0.5% and 1% of optimality for smaller and larger problems, respectively.

3.2 Metaheuristic Approaches

For methods based on metaheuristics, there is recent research in the literature reporting results on evolutionary programming (Juste et al, 1999), particle swarm optimization (Zhao et al, 2006), quantum evolutionary algorithms (Jeong et al, 2009; Lau et al, 2009), memetic algorithms (Valenzuela and Smith, 2002), and genetic algorithms (Kazarlis et al, 1996; Arroyo and Conejo, 2002; Sun et al, 2006; Dang and Li, 2007; Roque et al, 2011). Juste et al (1999) employ evolutionary programming in which populations of individuals evolve 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.

The resarch in (Zhao et al, 2006) introduces an improved particle swarm optimization (IPSO) where the orthogonal design is used for generating the initial population scattered uniformly over a feasible solution space. This method has been tested on the problems of case study 1 and presented good results. However, the method has been recently outperformed by both Jeong et al (2009) and Lau et al (2009).

These work propose Quantum-inspired Evolutionary Algorithms (QEAs). The QEA is based on the concept and principles of quantum computing, such as quantum bits, quantum gates and superposition of states. The QEA employs quantum bit representation, which has better population diversity comparatively to other representations used in evolutionary algorithms, and uses quantum gates to drive the population towards the best solution. The mechanism of the QEA can inherently treat the balance between exploration and exploitation, thus incorporating a sort of local search. Both Jeong et al (2009) and Lau et al (2009) divide the UC problem into two subproblems: 1) schedule the on/off status of the units and 2) determine the power output of the committed units. In both works, repair mechanisms are used to accelerate the solution quality and to ensure that unit schedules generated by the QEA are feasible. Jeong et al (2009) improve the conventional QEA 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, which have been improved in the work presented in this paper.

A Memetic Algorithm (MA) and a Genetic Algorithm (GA) using local search combined with Lagrangian relaxation are introduced in (Valenzuela and Smith, 2002). 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, it is only competitive for the latter. GA solutions to the UC problem have been provided in (Kazarlis et al, 1996) with the addition of problem specific operators. Problem spe-

cific operators are defined within windows, thus acting on building blocks rather than on bits. Therefore, once a good building block is found it is preserved through the evolution process.

Arroyo and Conejo (2002) propose a GA using a repair mechanism, which was implemented in parallel. Ramp rate limits are always enforced while constructing the solutions, and therefore they are 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 (2007) also divide the UC problem into scheduling and dispatching problems. The former is solved 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. The production of each on-line unit is determined by the LR. In (Sun et al, 2006) a real coded GA is proposed. A solution is represented by a real number matrix, expressing 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-unit problem (case study 2), being competitive only for the latter case study.

A very recent type of evolutionary algorithm, the Imperialist Competition Algorithm (ICA), has been applied to the UC problem in (Hadji and Vahidi, 2012). In it a population consists of a set of countries, all divided by imperialist countries and colonies, and based on the imperialistic power, which is inversely proportional to its cost function for a minimization problem. Then the colonies move toward their relevant imperialist country and the position of the imperialists is updated if necessary. In the next stage, the imperialistic competition among the empires begins, and in this competition, the weak empires are eliminated. The imperialistic competition will gradually lead to an increase in the power of dominant empires and a decrease in the power of weakest ones, until only one empire remains. The authors tested their methodology on the most commonly used benchmark problems (see case study 1). However, as demonstrated in the results section, the results only improve in the problem instance with 10 units.

More details on these methods and other applications developed for the UC problem can be found in the extensive and comprehensive bibliographic surveys published over the years, see for example Sen and Kothari (1998); Padhy (2000, 2004); Salam (2007).

4 The proposed methodology

In recent years many heuristic optimization approaches have been developed, one of the most popular being the GAs. Typically, the GAs develop several solutions as the result of selection, competition and recombination. Crossover and mutation are used to maintain a diversity of the evolving population and to escape from local optima. Several GAs have been proposed for the UC problem, see for example (Kazarlis et al, 1996; Arroyo and Conejo, 2002; Sun et al, 2006; Dang and Li, 2007; Abookazemi

et al, 2009). The GAs are a powerful stochastic global search technique as the search is performed by exploiting information sampled from different regions of the solution space (Reeves, 1993). Nevertheless, the 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, which means that the search is done within a pre-specified neighborhood. Including a Local Search procedure in a GA often leads to a substantial improvement since the "local" improvement capabilities of the former are being combined with the "global" nature of the GA. The GAs with random keys were first introduced by Bean (1994), for solving sequencing problems. In biased random key GAs, the bias is introduced at two different stages. Firstly, 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. Secondly, the crossover strategy is more likely to choose alleles from the best parent to be inherited by the offspring.

This work proposes a Hybrid Random Key Genetic Algorithm (HBRKGA), which is an improvement of the work in (Roque et al, 2011), based on the framework proposed by Gonçalves and Resende (2010). Here improved decoding and repair mechanisms are used. 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 natures for each of the violated constraints (Michalewicz and Janikow, 1991). In addition, to intensify the search around good solutions, a local search procedure was incorporated that, as confirmed in the results section, has led 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. In the first, solutions 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 that of the other parent. The HBRKGA framework is illustrated in Figure 1, an adaptation from (Gonçalves and Resende, 2010).

The decoding procedure and the fitness computation are specific to the problemaddressed. The decoding procedure, which is how solutions are constructed once a population of chromosomes is given, is performed in two main steps, as depicted 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 switched on according to their priority, which is given by the associated random key value. Furthermore, unit production is also set by a random key value. The production values are chosen such that the ramp rate constraints and the output range constraints are satisfied. Then, a second

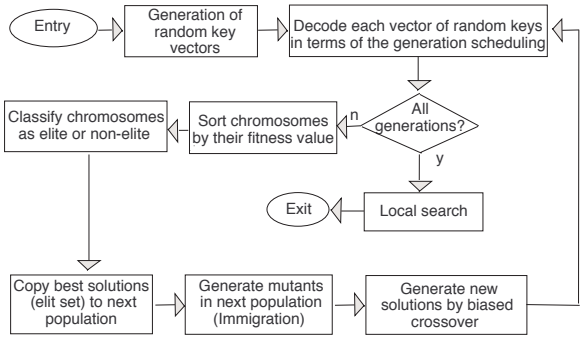

Fig. 1: The HBRKGA adapted framework.

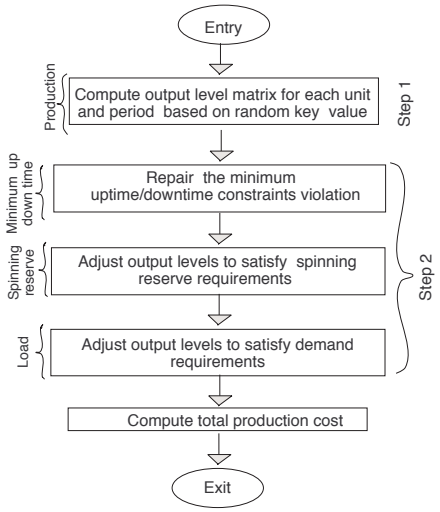

Fig. 2: Decoder flow chart.

step is applied since the solutions obtained may not be feasible. This step consists of verifying if the remaining constraints are met and of repairing the solutions whenever necessary.

4.1 Decoding Procedure

The decoding procedure proposed here is based on that of Roque et al (2011). The output generation levels are obtained based on the vector of random keys. Their values are computed such that the capacity limits and ramp rate limits are ensured during the decoding phase.

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 r_j, must be in the range defined by the minimum and maximum allowed output limits and ramp rate limits, as follows:

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

These limits are defined considering the unit output generation level limits and the ramp rate limits. At the same time, the ramp rate constraints are ensured for a specific time period t and 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) shows how these values are obtained.

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

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 is obtained using the repair mechanism illustrated in Figure 3. For two consecutive time 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.

For each period, the spinning reserve requirements may not be met. If the number of on-line units is not enough, some off-line units are switched on, one at a 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, which means in descending order of random key value. After ensuring that spinning reserve is met, there may be an excessive spinning reserve. Since this is not desirable due to the additional operational costs involved, it is necessary look for units that can be decommited. Units are switched off in ascending order of priority. At the end of this procedure, the U matrix is found, specifying which units are being operated at each time period, as well as the Y matrix, which indicates how much each on-line unit is producing. All constraints are satisfied except, perhaps, the load demand. Nevertheless, the maximum and minimum allowed production limits can be directly inferred from matrix Y. Therefore, it may be necessary to adjust the total production to meet load demand for each time period. Firstly, for all on-line units the production is set to its minimum value allowed. Next, for each time period, each unit is set to its maximum production allowed, one at a time, until the production reaches the load demand

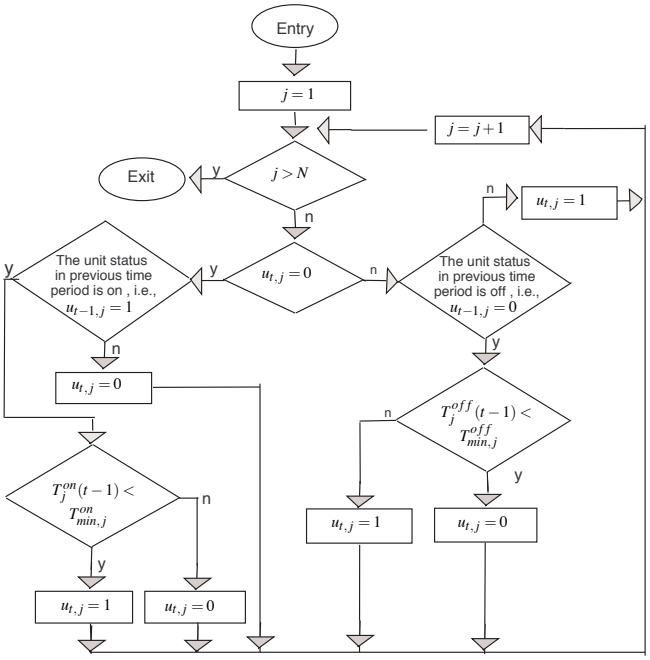

Fig. 3: Flowchart of Minimum up down time repair algorithm.

value. In doing so, units are considered in descending order of priority. 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, 3 subsets of solutions are combined as follows:

- Copied Solutions: 20% of the best solutions of the population of the current generation (elite set) are copied onto the next generation;
- Mutants: 20% of the solutions of the population of the next generation are obtained by randomly generating new 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.

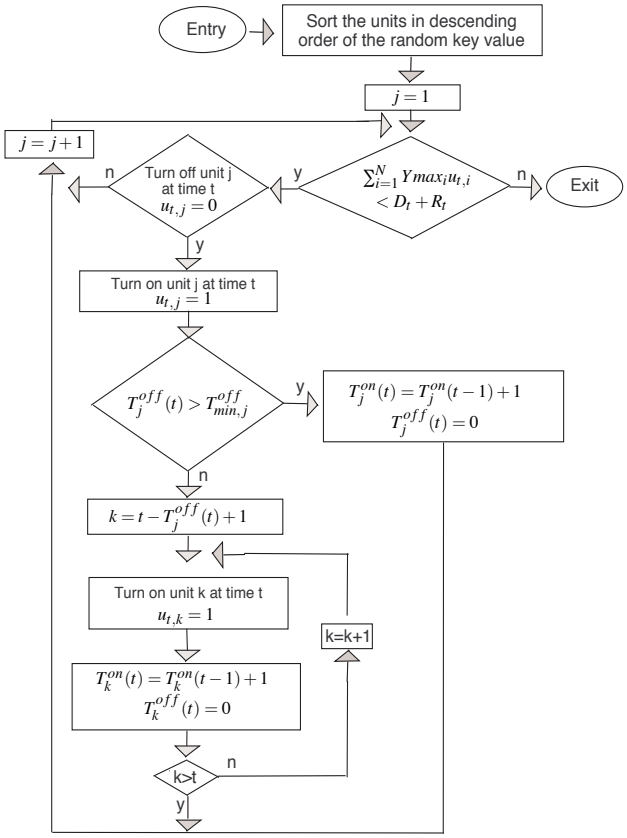

Fig. 4: Handling spinning reserve constraint.

As previously stated, 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. For the biased crossover, a biased coin is tossed to decide 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 this case).

4.3 Local Search

Another improvement to the previous work (Roque et al, 2011) is the inclusion of a local search procedure. At the end of the HBRKGA, a local search procedure is used 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 generating unit is replaced by an off-line generating unit if the swap is feasible and leads to a lower cost. Given a solution in the elite set, two sets of generating unit are built. Firstly, a set S_{on} containing the on-line generating units that can be turned off is built, and then a set S_{off} containing the off-line units that can be turned on.

For each time period, a pair of units is chosen, one from each of the sets built, and the feasibility of the swap is analyzed. If the swap is feasible, the total cost of the new solution is compared to 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, the next swap using the previously built sets is tried, which means that the sets S_{on} and S_{off} are not updated. 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 quality of the global solution can be seen in the results provided in Section 5.

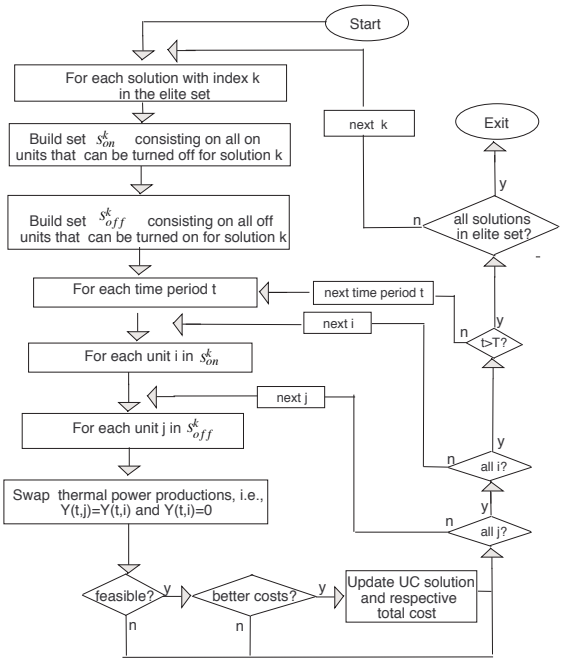

Fig. 5: Flow chart of the local search.

5 Numerical Results

This section presents the results obtained with the proposed HBRKGA, as well as the results obtained without using the local search, here referred to as BRKGA. It

should be noticed that the parameters are the same for both algorithms. Due to the stochastic nature of the methods proposed each problem was solved 20 times. Both GAs were implemented in Matlab. 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, case study 1 is singled out, since its problems are the ones that have been consistently considered in the literature and thus solved by many different methods and authors.

For comparison purposes, the UC problem is also formulated as a Mixed Integer Programming (MIP) model and solved using the commercial software CPLEX. The model is not derived here for the sake of simplicity. Instead, the reader is referred to Fuentes and Quintana (2002); Simoglou et al (2009), in which this model is based.

5.1 GA parameter setting

The present state-of-the-art theory on GAs does not provide information on how to configure the parameters involved in the algorithm. Therefore, the values used in the computational experiments conducted have been taken from the guidelines provided in (Ericsson et al, 2002; Gonçalves and Resende, 2010), as well as, from past experience (Roque et al, 2011).

Computational experiments with different values for the crossover probability, the number of generations, and the population size were conducted on the problem with 40 generating units provided in case study 1. The biased crossover probability was tested in 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 $5N$). For testing purposes the number of generations was set to a sufficiently large number ($NGers = 20N$).Soon it became clear that this number was too large, and therefore it was reduced to $10N$ (see Figure 6). To illustrate the algorithm behavior in Table 1 the results obtained are provided for varying P_c values with $NP = 2N$ and $NGers = 10N$. The value 0.7 is chosen since the best performances with lower variability correspond to this value. (Note that a better best solution was found using 0.6.)

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

In terms of the population size NP, as it can be confirmed in Figure 7, the quality of the solution is continuously and marginally improved with the population size, while the increase in computational time is almost linear. A trade-off analysis between the quality of the solution and the computational time led to set $NP = 2N$.

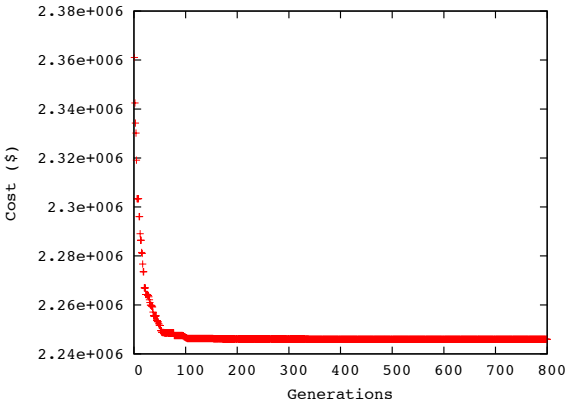

Fig. 6: Average cost for the 40-unit system (case study 1) for different number of generations.

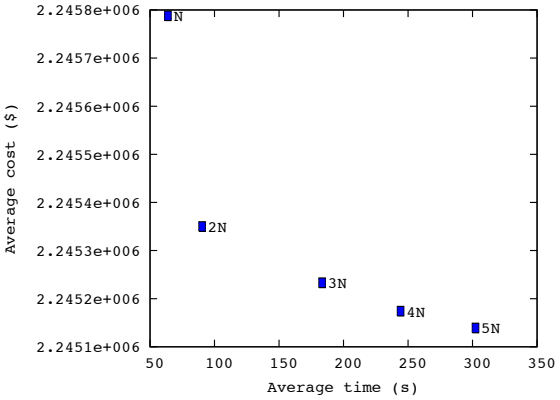

Fig. 7: Average cost and computational time for the 40-unit system (case study 1) for different population sizes.

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

In the following sections the results obtained by the BRKGA and the HBRKGA are compared to the best results reported in the literature. Furthermore, the CPLEX (version 12.1) was used to obtain an optimal solution and thus find how close the results are to the optimum. Nevertheless, such comparisons are only possible for small sized problems, since the CPLEX is unable to solve larger problems due to the large memory requirements. In addition, the CPLEX cannot handle the problems in case study 5 since the start-up costs are an exponential function of the number of hours that that unit has been down.

Table 2: Comparison between the best results obtained by the BRKGA and the HBRKGA and the best results reported in the literature for the problems in case study 1.

							CPLEX	HBRKGA		
Size	IPSO	IQEA	QEA	ICA	BRKGA	HBRKGA	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.

5.2 Case study 1

The HBRKGA and the BRKGA have been tested on a set of frequently used benchmark problems, involving systems with 10 up to 100 generating units and considering, in each case, a scheduling horizon of 24 hours. The 10 generating unit system, the base case, was originally proposed by Kazarlis et al (1996). Problem instances involving 20, 40, 60, 80 and 100 units are obtained by replicating the base case system and the load demands are adjusted proportionately to the system size. In all cases the spinning reserve is maintained at 10% of the hourly demand. The start-up costs have one of two possible values depending on the number of time periods the unit has been off, as given in equation (2). These values are different for each generation unit. The shut down costs are disregarded. Details of how these benchmark problems were constructed and on the system and demand data can be found in Kazarlis et al (1996).

For the problems in this case study, the CPLEX was capable of finding an optimal solution to systems involving 10 and 20 units. For problems with 40 units, the best solution found by the CPLEX is described. However, it crashed due to the excessive memory requirements. Although the solution is not optimal, it is the best solution found so far. In tables 2 to 4 it is possible to compare the results obtained (best, average, and worst) to the best former results (in italic) obtained with the different methods publish. The best current solution for each of the problems, excluding the CPLEX, is in bold. The last column, shows the gap between the HBRKGA solution and the previous best known solution. It should be noticed that whenever the HBRKGA produces a solution that is better than the best currently known solution the gap is negative. Table 2 reports on the optimality gap for the smaller problem instances, since for these the optimal solution value provided by the CPLEX). The results used for comparison purposes have been reported in: IPSO - Zhao et al (2006); IQEA - Jeong et al (2009); QEA - Lau et al (2009); ICA - Hadji and Vahidi (2012).

As observable in Table 2, for all problem instances, except one, our best results improve the best previously known results. Moreover, for the problem instances for which an optimal solution has been found by the CPLEX, it can be confirmed that the HBRKGA found an optimal solution in one case, while in the other case the solution found is within 0.06% of optimality. By comparing the HBRKGA and the BRKGA, which already improves some of the previously known best solutions, it is possible

Table 3: Comparison between the average results obtained by the BRKGA and the HBRKGA and the best average results reported in the literature for the problems in case study 1.

					HBRKGA	
Size	IQEA	QEA	BRKGA	HBRKGA	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 between the worst results obtained by the BRKGA and the HBRKGA and the best worst results reported in the literature for the problems in case study 1.

						HBRKGA	
Size	IPSO	IQEA	QEA	BRKGA	HBRKGA	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

to verify that the local search is always effective since the HBRKGA is always better than the BRKGA. And the improvement ranges from 0.007% to 0.063%. Although these values are small their impact is relevant as they refer to a multi-million dollar industry.

The average results have also improved for all but one of the problem instances solved, when compared to the best previously known results (see Table 3). Table 4 provides similar results for the worst solutions. Again, the best previous results have been improved. The results reported in these tables also show that the local search incorporation is effective, since the HBRKGA improves upon the BRKGA.

Another important feature of the proposed algorithm is that, as it can be seen in Table 5, the variability of the results is quite small. The difference between the worst and the best solutions found for each problem is always below 0.14%; however, if the best and the average solutions are compared this difference is never larger than 0.05%. This makes it possible to infer on the robustness of the approach, which is very important since the industry is reluctant to use methods with high variability as this may lead to using poor solutions. When compared to the robustness of the alternative methods, the proposed approached is better than that of the IPSO and the QEA and almost the same as that of the IQEA.

In terms of computational time, no exact comparisons are possible because not only are the values obtained on different hardware, but also because the HBRKGA reported time is real time and not CPU time and thus it is not directly comparable to others reported in the literature. The computational experiments were performed on a Xeon X5450, 3.0 GHz and 4.0 GB RAM. This is a shared machine and therefore

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

Size	$\frac{Average-Best}{Best}\%$			$\frac{Worst-Best}{Best}\%$				St. deviation(%)		
	HBRKGA	IQEA	QEA	HBRKGA	IPSO	IQEA	QEA	HBRKGA	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

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

Size	IPSO	IQEA	QEA	ICA	BRKGA	HBRKGA	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	–

several processes are usually running in parallel. Nevertheless, Table 6 shows the computational time requirements, as well as the works used for comparison purposes. It should be highlighted that the results reported for the IPSO and the IQEA may not be accurate since the authors only provide them in a graphical form. These results are also provided graphically in Figure 8. As it can be confirmed, the IPSO presents computational time requirements which are much larger than the other methods. On the contrary, the QEA is the fastest method. The other three methods have a similar behavior in terms of computational requirements. Therefore, the HBRKGA presents an intermediate performance, regarding computational time, which is not a big price to pay for the increased quality of the solution quality. Recall that, as confirmed in Table 2, the HBRKGA provides the best solution for all but one of the problems analyzed in this case study.

When the computational time is analyzed in a logarithmic scale, as in the graph on the right-hand side of Figure 8, a favorable conclusion can be drawn on the algorithms proposed here. 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 contrast, algorithms proposed here present a concave growth in the log scale, meaning that the time increase is subexponential.

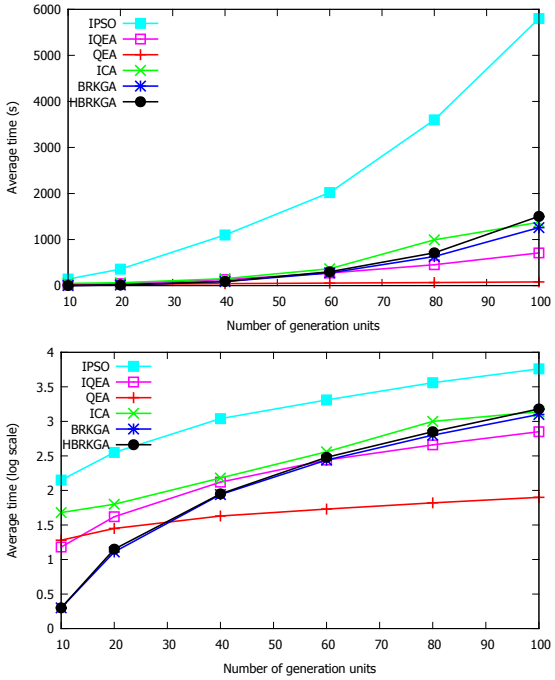

Fig. 8: Computational time requirements, in seconds and log scale, for the methods being compared, for the problems in case study 1.

5.3 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 Huang et al (1998). The start up costs are constant, not necessarily different for all units, while the shut down costs are disregarded. The spinning reserve is set to 11% of hourly load and ramp rate constraints are also taken into consideration. The characteristics of the generating units, the load demand, and the specific conditions of the problem are given in Huang et al (1998). This specific problem has not been considered by many authors conducting research on the UC problem. Therefore, the approach prosented is compared to the four approaches proposed in Huang et al (1998), which are based on dynamic programming (DP), Lagrangian Relaxation (LR), Simulated Annealing (SA), and Constraint Logic Programming (CLP), and also to a GA (MRCGA) recently proposed by Sun et al (2006). In addition, the solutions are compared to the solution obtained by the CPLEX. This solution may not be optimal since the CPLEX has not run until the end due to the excessive memory requirements. However, the best solution found, before the CPLEX crashed, is referred, as well as the time it took to find such a solution for the first time.

Both the BRKGA and the HBRKGA improve the best known solutions for all cases (best, average and worst). Again the local search has proved to be effective

Table 7: Comparison between the results obtained by the BRKGA and the HBRKGA and the best results reported in the literature for the problems in case study 2.

Size	DP	LR	CLP	MRCGA	BRKGA	HBRKGA	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

since in all cases the HBRKGA obtains better solutions than the BRKGA. The computational times are not a concern since the method that takes longer (the DP by Huang et al (1998)) requires just over 3 minutes.

5.4 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. Only start-up costs are considered and they are constant, although they are not necessarily the same for all units. The spinning reserve requirement is set at 400MW for each time period. The system and demand data can be found in Venkatesh et al (2007), as well as, the conditions used in the computational experiments. The quality of the solution obtained by the BRKGA and by the HBRKGA is compared to that of the fuzzy mixed integer Linear Programming proposed in Venkatesh et al (2007). It was possible to find an optimal solution to this problem by using the CPLEX. As seen in Table 8, both the BRKGA and the HBRKGA improve the previously best known results. Again, the use of the local search made it possible to obtaining an improved solution. Furthermore, the best solution obtained by the HBRKGA is very close to an optimal solution. Thus, the CPLEX cannot be considered a better alternative when compared to the HBRKGA since the latter obtains a solution within 0.26% of optimality, being about 21 times faster.

Table 8: Comparison between the results obtained by the BRKGA and the HBRKGA and the best results reported in the literature for the problems in case study 3.

Size	FMILP	BRKGA	HBRKGA	CPLEX
Best	*722388*	722260	**721197**	719314
Average	–	722283	**721202**	–
Worst	–	722410	**721212**	–
Gap (%)	0.43	0.41	0.26	–
St.deviation(%)	–	0.01	0.01	–
Av.Time(s)	25.5	24.3	29.6	642

5.5 Case study 4

The problem addressed in this case study comprises 45 units and a planning horizon of 24 hours. The system data and the load demand can be found in (Arroyo and Conejo, 2002). The spinning reserve is set to 10% of the load demand at every hour. Both the start-up and the shut-down costs are constant, although not necessarily the same for all units. Table 9 shows the best solutions known so far, obtained in Arroyo and Conejo (2002) from three versions of a GA: global parallelization (GP), which uses a parallel implementation of the repair algorithm, a coarse-grained parallel genetic algorithm (CGPGA), which evolves several populations independently, one in each processor, and a hybrid parallel genetic algorithm (HPGA), which combines both previous parallelizations. In addition, the best, average and worst solutions are reported for both the BRKGA and the HBRKGA. The methods proposed here improve the best known solution by 0.22%. For this problem the local search was not effective since the cost of the best, average and worst solutions are the same for HBRKGA and the BRKGA. It should be highlighted that the CPLEX was unable to provide any solution for this problem due to its size.

Table 9: Comparison between the results obtained by the BRKGA and the HBRKGA and the best results reported in the literature for the problems in case study 4.

Size	GP	CGPGA	HPGA	BRKGA	HBRKGA
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

In terms of the computational time, although the approaches presented here have not been implemented in parallel, they are faster than the approach producing the best former results.

5.6 Case study 5

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

Both problems in this case study involve the scheduling of 10 units over a 24-hour time horizon. In both cases the shut-down costs are disregarded.

In the first problem, the spinning reserve is set to 10% of the hourly load demand. All problem data are provided in (Turgeon, 1978), where it has been first addressed. The start-up costs are computed as:

$$S_{t,j} = b_0 . \left(1 - b_1 . e^{-b_2 t}\right). \tag{11}$$

This problem has been addressed in Valenzuela and Smith (2002) where an optimal solution has been found by using dynamic programming. 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).

Table 10 shows the results published in (Valenzuela and Smith, 2002), as well as the results obtained by the approaches suggested here. It was possible 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 (Valenzuela and Smith, 2002). However, the LR was capable of finding a better solution. In terms of computational time, the methodologies suggested perform much better as they are up to 53 times faster. For this problem, again the local search does not help find a better solution.

Table 10: Comparison between the results obtained by the BRKGA and the HBRKGA and the best results reported in the literature for the first problem in case study 5.

Size	DP	LR	GA	MA	LRMA	BRKGA	HBRKGA
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 has been proposed in (Bard, 1988), where the problem data can be found. The spinning reserve requirements are specified for each time period and vary between 6.47% and 11.35%. The start-up costs depend exponentially on the number of time periods during which the unit has been off. The start-up costs are given as follows:

$$S_{t,j} = b_0 . \left(1 - e^{- \frac{max(0, -T_j^{off}(t))}{b_2}} \right) + b_1 . \tag{12}$$

More recently, other authors have addressed this problem. Table 11 compares the results achieved with those obtained by the LR due to Bard (1988), and the recently proposed heuristics: DP - (Valenzuela and Smith, 2002); MA - (Valenzuela and Smith, 2002); FPGA - (Dang and Li, 2007).

As it is possible to confirm, neither the heuristics proposed recently nor the algorithms proposed here were able to improve the best known results found by the LR due to Bard (1988). The HBRKGA is the method that provides the best results in terms of the quality of the average and worst solutions. It is important to highlight that the BRKGA also presents better average and worst results than the other heuristics. Therefore, the BRKGA and the HBRKGA methods present solutions with the lowest variability. Moreover, the BRKGA and the HBRKGA average execution times

Table 11: Comparison between the results obtained by the BRKGA and the HBRKGA and the best results reported in the literature for the second problem in case study 5.

Size	DP	LR	MA	FPGA	BRKGA	HBRKGA
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

are significantly shorter than those of the other methods, with the HBRKGA being up to 43 times faster than the DP heuristic. The local search is effective for this problem since the quality of the HBRKGA solution is better for all solution types.

6 Conclusions

The unit commitment problem is an important area of research which has attracted increasing interest from the scientific community due to the fact that small savings in the operation costs for each hour can lead to major overall economic savings.

Biased Random Key GAs have been developed for and applied to several combinatorial optimization problems with interesting results. Given this empirical evidence, see (Gonçalves and Resende, 2010), this paper proposes an algorithm for the unit commitment problem. The Hybrid Biased Random Key Genetic Algorithm proposed here also incorporates a Local Search to intensify the search near good solutions. The approach suggested in this paper searches for a solution only within the space of feasible solutions using a repair mechanism.

The performance of the algorithm implementing the proposed approach has been tested on a set of commonly used UC benchmark problems and other UC problems found in the literature. The results reported show that the method proposed here outperforms current state-of-the-art methods. For all problem instances but two, it was possible to find better results than the best achieved thus far. In addition, these better solutions have been found with computational time requirements which are smaller or of the same magnitude as those in the alternative methods. The results also show another important feature, which is lower variability. It is important to highlight that for the most commonly used problems 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%. This is very important since the methods to be used in industrial applications must be robust in order to guarantee that poor solutions are not implemented.

Acknowledgements We acknowledge the support of the ERDF (FEDER), the COMPETE through the FCT as part of projects PTDC/EGE-GES/099741/2008 and PTDC/EEA-CRO/116014/2009 and the North Portugal Regional Operational Programme (ON.2 O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

References

Abookazemi K, Mustafa M, Ahmad H (2009) Structured Genetic Algorithm Technique for Unit Commitment Problem. International Journal of Recent Trends in Engineering 1(3):135–139

Arroyo J, Conejo A (2002) A parallel repair genetic algorithm to solve the unit commitment problem. IEEE Transactions on Power Systems 17:1216–1224

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

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

Carrion M, Arroyo J (2006) A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. IEEE Transactions on Power Systems 21(3):1371–1378

Cohen AI, Yoshimura M (1983) A Branch-and-Bound Algorithm for Unit Commitment. IEEE Transactions on Power Apparatus and Systems 102(2):444–451

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

Ericsson M, Resende M, Pardalos P (2002) A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. Journal of Combinatorial Optimization 6(3):299–333

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

Fontes DB, Gonçalves JF (2007) Heuristic solutions for general concave minimum cost network flow problems. Networks 50(1):67–76

Fontes DB, Gonçalves JF (2012) A multi-population hybrid biased random key genetic algorithm for hop-constrained trees in nonlinear cost flow networks. Optimization Letters pp 1–22

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

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

Frangioni A, Gentile C, Lacalandra F (2009) Tighter Approximated MILP Formulations for Unit Commitment Problems. IEEE Transactions on Power Systems 24(1):105–113

Michalewicz Z, Janikow C (1991) Semidefinite Programming: A Practical Application to Hydro-Thermal Coordination. In Proceedings of the Fourteenth International Power Systems Computation Conference (PSCC), Seville, Spain.

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

Gonçalves J, Resende M (2011) A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. Journal of Combinatorial Optimization 22(2):180–201

Gonçalves J, Mendes JM, Resende M (2008) A genetic algorithm for the resource constrained multi-project scheduling problem. European Journal Operational Re-

search 189(3):1171–1190
Hadji M, Vahidi B (2012) A Solution to the Unit Commitment Problem Using Imperialistic Competition Algorithm . IEEE Transactions on Power Systems 27(1):117–124
Huang K, Yang H, Yang C (1998) A new thermal unit commitment approach using constraint logic programming. IEEE Transactions on Power Systems 13(3)
Jeong Y, Park J, Shin J, Lee K (2009) A thermal unit commitment approach using an improved quantum evolutionary algorithm. Electric Power Components and Systems 37(7):770–786
Jiang R, Wang J, Guan Y (2012) Robust unit commitment with wind power and pumped storage hydro. IEEE Transactions on Power Systems 27(2):800–810
Juste K, Kita H, Tanaka E, Hasegawa J (1999) An evolutionary programming solution to the unit commitment problem. IEEE Transactions on Power Systems 14(4):1452–1459
Kallrath J, Pardalos P, Rebennack S, Scheidt M (2009) Optimization in the Energy Industry . Energy Systems, Springer
Kazarlis S, Bakirtzis A, Petridis V (1996) A Genetic Algorithm Solution to the Unit Commitment Problem. IEEE Transactions on Power Systems 11:83–92
Kotsireas I, Koukouvinos C, Pardalos P, Simos D (2012) Competent genetic algorithms for weighing matrices. Journal of Combinatorial Optimization 24(4):508–525
Lau T, Chung C, Wong K, Chung T, Ho S (2009) Quantum-Inspired Evolutionary Algorithm Approach for Unit Commitment. IEEE Transactions on Power Systems 24(3):1503–1512
Lauer G, Sandell N, Bertsekas D, Posbergh T (1982) Solution of large scale optimal unit commitment problems. IEEE Transactions on Power Apparatus and Systems PAS-101(1):79–96
Michalewicz Z, Janikow C (1991) Handling Constraints in Genetic Algorithms. In: In Belew, R.K., Booker, L.B., eds.: Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91), San Mateo, California, University of California, San Diego, Morgan Kaufmann Publishers, pp 151–157
Muckstadt J, Koenig S (1977) An application of Lagrangian relaxation to scheduling in power-generation systems . Operations Research 25(3):387–403
Ostrowski J, Anjos MF, Vannelli A (2012) Tight mixed integer linear programming formulations for the unit commitment problem. IEEE Transactions on Power Systems 27(1):39–46
Padhy N (2000) 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
Padhy N (2004) Unit Commitment-A Bibliographical Survey. IEEE Transactions on Power Systems 19(2):1196–1205
Patra S, Goswami S, Goswami B (2009) Fuzzy and simulated annealing based dynamic programming for the unit commitment problem. Expert Systems with Applications 36(3):5081–5086
Rebennack S, Pardalos P, Pereira MV, Iliadis N (2010a) Handbook of Power Systems I. Energy Systems, Springer

Rebennack S, Pardalos P, Pereira MV, Iliadis N (2010b) Handbook of Power Systems II. Energy Systems, Springer

Reeves CR (1993) Modern Heuristic Techniques for Combinatorial Problems. chapter Genetic Algorithms, Blackwell Scientific Publications

Rong A, Hakonen H, Lahdelma R (2008) A variant of the dynamic programming algorithm for unit commitment of combined heat and power systems. European Journal Operational Research 190:741–755

Roque L, Fontes DBMM, Fontes FACC (2011) A Biased Random Key Genetic Algorithm Approach for Unit Commitment Problem. Lecture Notes in Computer Science 6630(1):327–339

Salam S (2007) Unit commitment solution methods. In: Proceedings of World Academy of Science, Engineering and Technology, vol 26, pp 600–605

Schneider F, Klabjan D, Thonemann U (2013) Incorporating demand response with load shifting into stochastic unit commitment. Available at SSRN 2245548

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

Simoglou CK, Biskas PN, Bakirtzis AG (2010) Optimal self-scheduling of a thermal producer in short-term electricity markets by MILP. IEEE Transactions on Power Systems 25(4):1965–1977

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

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

Turgeon A (1978) Optimal scheduling of thermal generating units. IEEE Transactions on Automatic Control 23:1000–1005

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

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

Viana A, Pedroso J (2013) A new MILP-based approach for unit commitment in power production planning. Electrical Power and Energy Systems 44(1):997–1005

Wang Q, Guan Y, Wang J (2012) A chance-constrained two-stage stochastic program for unit commitment with uncertain wind power output. IEEE Transactions on Power Systems 27(1):206–215

Zhao B, Guo C, Bai B, Cao Y (2006) An improved particle swarm optimization algorithm for unit commitment. International Journal of Electrical Power & Energy Systems 28(7):482–490

Zheng Q, Wang J, Pardalos P, Guan Y (2012) A decomposition approach to the two-stage stochastic unit commitment problem. Annals of Operations Research pp 1–24