Turkish Journal of Electrical Engineering and Computer Sciences

Volume 16 | Number 3 Article 7

1-1-2008

Evolutionary Algorithms for the Unit Commitment Problem

A. ŞİMA UYAR

BELGİN TÜRKAY

Turk J Elec Engin, VOL.16, NO.3 2008, © TÜBİTAK

Evolutionary Algorithms for the Unit Commitment Problem

A. Şima UYAR[1], Belgin TÜRKAY[2]
[1] *Department of Computer Engineering, Faculty of Electrical and Electronics, İstanbul Technical University, Maslak, 34469 İstanbul-TURKEY*
e-mail: etaner@itu.edu.tr
[2] *Department of Electrical Engineering, Faculty of Electrical and Electronics, İstanbul Technical University, Maslak, 34469 İstanbul-TURKEY*
e-mail: turkayb@itu.edu.tr

Abstract

This paper compares three evolutionary computation techniques, namely Steady-State Genetic Algorithms, Evolutionary Strategies and Differential Evolution for the Unit Commitment Problem. The comparison is based on a set of experiments conducted on benchmark datasets as well as on real-world data obtained from the Turkish Interconnected Power System. The results of two state-of-the-art evolutionary approaches, namely a Generational Genetic Algorithm and a Memetic Algorithm for the same benchmark datasets are also included in the paper for comparison. The tests show that Differential Evolution is the best performer among all approaches on the test data used in the paper. The performances of the other two evolutionary algorithms are also comparable to Differential Evolution and the results of the algorithms taken from literature showing that all EA approaches tested here are applicable to the Unit Commitment Problem. The results of this experimental study are very promising and promote further study.

Key Words: *Unit commitment problem, evolutionary algorithms, steady-state genetic algorithms, generational genetic algorithms, differential evolution, evolutionary strategies.*

1. Introduction

The Unit Commitment Problem (UCP) is a constrained optimization problem in which optimal turn-on and turn-off schedules need to be determined over a given time horizon for a group of power generation units under some operational constraints. The objective is to minimize the power generation costs while meeting the hourly forecasted power demands. The UCP is an important area of research which has attracted increasing interest from the scientific community due to the fact that even small savings in the operation costs for each hour can lead to major overall economic savings.

The UCP consists of two sub-problems which are usually solved separately. First the schedule for the turn-on and turn-off times of the power generation units is determined. Then, based on this schedule, the objectives and the constraints, the amounts of power to be generated by the online generators for each hour are calculated. Several approaches exist in literature to tackle the UCP, such as dynamic-programming [1,

2], Lagrangian relaxation and heuristics with hybrids [3–6], branch and bound [7], benders decomposition [8], simulated annealing [9], tabu-search [10], evolutionary algorithms [11–17] and many hybrids. A detailed survey can be found in [18–20].

Evolutionary Algorithms (EAs) [21] are population-based optimization techniques based on the classical Mendelian laws of inheritance and Darwin's theory of evolution. EA is an umbrella term that covers several approaches, namely Genetic Algorithms (GA) [22], Evolutionary Strategies (ES) [23], Genetic Programming (GP) [24] and Differential Evolution (DE) [25] which are based on the same principles but differ in the application of these principles. EAs have been successfully used in many problem domains. GAs, ES and DE are mainly for search and optimization while GP is used more for automatic program generation, prediction and machine learning tasks. The solution to the UCP is given as a set of binary decision variable assignments showing which generator units are online and which are offline for any given time slot. This solution is obtained through minimizing a cost objective while adhering to several constraints. Therefore this problem can be seen as a search for feasible solutions which optimize an objective. Due to this, the performance of only GAs, ES and DE is explored in this paper and GP is left out.

In [26] initial tests using only the DE approach were performed by the authors and promising results were obtained. This study furthers the previous work by exploring the behavior of three major EA implementations for the UCP. These three EA variants are tested on benchmark UCP data as well as on real-world data of the Turkish Interconnected Power System. The results are also compared to successful results reported in literature for the same benchmark datasets.

This paper is organized as follows: In section 2, the UCP is explained. Section 3 introduces the EA approaches used in this study. In section 4, experiments and results are given. Section 5 concludes the paper.

2. The Unit Commitment Problem

The objective of the UCP is to minimize the total cost of power generation over a given time horizon subject to operational constraints while satisfying hourly power demands. Two main factors affect the cost: fuel costs and start-up costs. There are two types of constraints: constraints based on the operational restrictions of the generators and the constraints based on power generation requirements. The parameters used in the UCP formulation are given in Table 1.

Table 1. Parameters used in the UCP formulation.

$P_i(t)$	Power generated by unit $i$at time t
$F_i(p)$	Cost of producing pMW power by unit I
P_i^{max}	Maximum power which can be generated by unit I
P_i^{min}	Minimum power which can be generated by unit I
$PD(t)$	Power demanded at time t
$PR(t)$	Power determined as reserve at time t
$CS_i(t)$	Start-up cost of i^{th} unit at time t
$x_i(t)$	Duration that unit $i$has not changed its status
$v_i(t)$	Status of i^{th} unit at time t(online/offline)

Fuel cost depends on the amount of power output provided by each online unit for each time slot. The fuel cost needs to be minimized subject to two constraints: the power demands for each hour should be met and the power generated by each unit should be within its minimum and maximum capacities. This

part of the objective can be formulated as

$$\min F_{total}(t) = \sum_{i=1}^{N} F_i(P_i(t))$$

subject to the constraints

$$\sum_{i=1}^{N} F_i(P_i(t)) = PD(t)$$

$$P_i^{\min} \leq (t) \leq P_i^{\max}$$

Start-up costs depend on the number of hours a unit has been offline. The formulation for the start-up cost is

$$CS_i \begin{cases} CS_{\text{hot}} & \text{if} \quad x_i(t) \leq t_{\text{coldstart}} \\ CS_{\text{cold}} & \text{otherwise} \end{cases}$$

where $t_{coldstart}$ is a parameter which defines a threshold for a hot or a cold start-up specific to each generation unit type. An additional constraint (*minimum up/down time)* defines a minimum amount of time for each generator during which the generator is not allowed to change its status. The formulation for these constraints is

$$\begin{array}{lll} \text{if} & v_i(t) = 1 & x_i(t-1) \geq t_{\text{down}} \\ & \text{else} & x_i(t-1) \geq t_{\text{up}} \end{array}$$

According to these fuel cost and start-up cost functions and constraints, the formulation for the UCP for N units and T hours can be summarized as

$$\min \ F_{\text{total}} = \sum_{t=1}^{T} \sum_{i=1}^{N} [F_i(P_i(t)).v_i(t) + CS_i(t)]$$

subject to constraints

$$\sum_{i=1}^{N} P_i(t).v_i(t) = PD(t)$$

$$v_i(t).P_i^{\min} \leq P_i(t) \leq v_i(t)P_i^{\max}$$

$$\sum_{i=1}^{N} P_i^{\max}(t).v_i(t) \geq PD(t) + PR(t)$$

$$\begin{array}{lll} \text{if} & v_i(t) = 1 & x_i(t-1) \geq t_{\text{down}} \\ & \text{else} & x_i(t-1) \geq t_{\text{up}} \end{array}$$

The fuel cost of generating pMW of power for the i^{th} unit is calculated using the formula

$$F_i(p) = a_{0i} + a_{1i}.p + a_{2i}.p^2$$

The fuel generation cost depends on three system parameters: a_{0i}, a_{1i} and a_{2i}. These parameters are given for each generator i. After a schedule has been found, the power to be generated by each online

generator needs to be determined. This part of the UCP is called the *economic dispatch problem* (EDP) [27] and is commonly solved using *lambda-iteration* [10, 15]. We also use this method in our experiments. The lambda-iteration algorithm is given in Figure 1.

```
select initial λ and Δ;
repeat
      calculate P_i for each generator using dF_i/dP_i = λ;
      calculate P_total;
      diff=PD-P_total;
      if (diff<0) then
            λ=λ-Δ;
      else
            λ=λ+Δ
      endif;
      Δ=Δ/2;
until (|diff|<ε);
```

Figure 1. Lambda-Iteration Algorithm.

The initial value of λ and Δ are determined as given below, where $\lambda_{\max}$ and $\lambda_{\min}$ are calculated using $P_{\max}$ and $P_{\min}$:

$$\lambda = \frac{\lambda_{\max} + \lambda_{\min}}{2}$$

$$\lambda = \frac{\lambda_{\max} - \lambda_{\min}}{2}$$

3. Evolutionary Algorithms

EAs [21] have been applied to many different problem domains with successful results. All variants of EAs are based on the same natural principles; however they mainly differ in the solution representations and operators used and sometimes in the order in which these operators are applied. The algorithmic flow of a basic EA is given in Figure 2.

```
select initial population;
repeat
      evaluate population;
      select mating pairs;
      apply reproduction operators;
until stopping_criteria satisfied;
```

Figure 2. Basic Evolutionary Algorithm.

EAs are population-based search and optimization techniques. They start from an initial set of candidate solutions in the search space termed as the *initial population*. Each parameter of the problem is called a *gene* and the solution string made up of the parameters is a *chromosome*. A solution candidate to the problem is termed as an *individual* and commonly consists of only one chromosome.

The initial population is usually determined randomly. After the initialization step, the main loop of the EA begins. In each loop, first the current set of individuals is evaluated based on the objectives and the constraints. The function used to evaluate these individuals is termed as the *fitness function*. Individuals with high fitness values are those that produce solutions of higher quality. In the second step of the EA loop, some of the individuals are selected to undergo reproduction. According to the technique used to select these individuals, higher quality solutions have a higher chance of being selected, thus transmitting their good genetic material into the next generations.

In the reproduction stage, two genetic operators are applied: *crossover* and *mutation*. Crossover acts on two individuals termed as *parents*, creating a new individual, the *offspring*, by combining parts of the chromosomes from each parent. The resulting offspring individual then undergoes mutation which causes changes in the solution represented by the individual. Crossover aids in exploiting already found good solutions, while mutation allows the EA to explore different parts of the search space.

After the required number of offspring has been generated through selection and reproduction, some of these individuals are selected to form the population in the next *generation* (iteration). Different techniques are employed by different EA variants at this stage. Commonly, the *population size*, i.e. the number of individuals in the population, in each generation is kept constant in most EAs.

The EA loop continues until some stopping criteria are met. While several criteria exist in literature, the most common form is allowing the EA to explore a limited amount of solution candidates in the search space. Since each solution candidate is evaluated using the fitness function, this criterion is usually implemented as allowing the EA to run until a predetermined *number of fitness evaluations* has been performed. Further details on EAs can be found in [21].

3.1. Genetic algorithms

GAs are among the earlier variants of EAs, developed by Holland [22] in 1975. GAs commonly work on a binary representation of solutions and thus are very suitable for the UCP. There are some existing applications of various GAs to the UCP in literature, e.g. as in [11–17].

In GAs, there are two most commonly used population replacement schemes: the *generational replacement scheme* and the *steady-state replacement scheme*. GAs using generational-replacement are called *generational GAs* (GGAs) and those that use the steady-state model are called *steady-state GAs* (SSGAs). In GGAs, in each iteration the whole population is replaced. However in SSGAs, only one offspring is generated in each iteration of the EA loop and the new individual replaces an existing one in the population. By doing this, the population size is kept constant.

In the GA applications of UCP in literature, commonly a GGA model is used. Therefore, to explore whether SSGA is also suitable, the steady-state model is adopted in this paper. Therefore details of SSGA will be further given here. The SSGA follows the basic EA flow but at each iteration only one offspring is generated and inserted into the population. The basic SSGA algorithm is given in Figure 3.

```
randomly select initial population;
repeat
      select one mating pair;
      generate one offspring through reproduction;
      evaluate offspring;
      if offspring better than current worst individual then
      offspring replaces worst individual;
      endif
until max_fitness_evaluations reached;
```

Figure 3. Basic Steady State Genetic Algorithm.

For mating pair selection, *binary tournament selection* is used. The two selected individuals go through reproduction, which consists of crossover and mutation. A *two-point crossover* method is used, in which the offspring gets a segment of its solution, defined by two cutoff points, from one parent while the rest is taken from the other. Crossover occurs with a predefined *crossover probability* (pc). If crossover does not occur between the parents, the new offspring becomes an exact copy of one of the parents which is randomly determined. *Point mutation* is used on the offspring generated as a result of crossover. In this type of mutation, the value of a parameter on the solution string is inverted with a predefined *mutation probability* (pm). After the fitness value of the offspring is calculated, it is compared to the fitness of the worst individual in the current population. If the offspring has a better fitness, it replaces the worst individual, otherwise it is discarded. The loop continues until a fixed amount of fitness evaluations have been performed by the SSGA. Further details can be found in [21].

3.2. Evolutionary strategies

Evolutionary Strategies (ES) was introduced by Rechenberg in 1973 [23]. ES is traditionally applied to continuous search spaces. The UCP is a discrete problem; therefore a modified version of ES is used in this paper. Similar to GAs, ES also works on a population of individuals however the application of the genetic operators is somewhat different. The general outline of a basic ES is given in Figure 4.

In ES, a chromosome consists of three parts: The genes corresponding to the parameters of the problem form the first part. In the second part, the mutation step sizes associated with each gene are located. The third part gives the rotation angles used in calculating the mutation step sizes. Commonly, only the first two parts are used and the third part is ignored. So a sample chromosome in ES is as follows:

$$< \mathrm{p}_1\ \mathrm{p}_2 \cdots \mathrm{p}\ ,\ \ \sigma_1\ \sigma_2 \cdots \sigma_n >$$

where p_i show the parameters and the σ_i show the corresponding mutation step sizes.

```
randomly select initial population;
evaluate population;
repeat
      for child_population_size times do
         randoly select one mating pair;
         generate one offspring through reproduction;
      endfor
      select population_ size individuals;
until max_fitness_evaluations reached;
```

Figure 4. Basic Evolutionary Strategies Algorithm.

In ES, in each iteration, μ number of offspring is generated from λ individuals through reproduction. μ is commonly chosen to be much larger than λ. In ES, parent selection is done randomly. For each offspring, two parent individuals are selected uniformly randomly from the whole population. These individuals first go through crossover then through mutation. Two different types of crossover are applied to the two different parts of the chromosome. In the parameters part, the same types of crossover methods as in GAs are commonly used. In this study, a two-point crossover technique is chosen as in the SSGA. For the crossover in the second part of the chromosome, where the mutation step sizes are located, an arithmetic averaging is commonly performed. This approach is also chosen in this study.

Self-adaptive mutation is a feature of ES. Each parameter has its own mutation step size which is also adapted through evolution. *Gaussian mutation* is applied to mutate both the parameters and the mutation step sizes. The mutation step sizes are mutated first. Then these are used as the standard deviation values in the normal distribution for the Gaussian mutation of the parameters. Through this process, good mutation values which generate good solutions are preserved on the chromosomes.

In ES one of two different methods may be used to determine the individuals for the next generation. In the *plus-strategy*, the best λ individuals from the parents and offspring combined are selected. In the *comma-strategy*, selection is done from only the offspring. The comma-strategy is better at preserving diversity, therefore it is chosen in this paper. Again the loop continues until a fixed amount of fitness evaluations have been performed by the ES. Detailed information on ES can be found in [28].

In order to apply ES to the UCP, some modifications are needed. Since the gene values are binary, Gaussian mutation is not applicable. In the binary version of ES, the second part of the chromosome denotes the mutation probability pmi of the ith gene rather than a mutation step size. This means that a gene is mutated, i.e. inverted, with a probability given by the corresponding mutation probability value in the second part of the chromosome. The mutation probability values themselves are still mutated using Gaussian mutation. The rest of the ES flow and operations are the same as in the continuous case.

3.3. Differential Evolution

The Differential Evolution (DE) [25] algorithm was introduced by Storn and Price in 1995. DE is also a form of an evolutionary algorithm which operates in continuous search spaces. DE is based on four main steps: Initialization, mutation, recombination and selection. All individuals pass through these operations. In DE

literature, the chromosome is commonly referred to as *the vector*. The same terminology is also adopted in this paper. The algorithmic flow of the basic DE is given in Figure 5.

The chromosomes of an individual are made up of real valued genes each of which correspond to the parameters of the problem to be optimized. All individuals in the population, called the *target vectors*, go through the mutation and recombination steps. There are several mutation operators. In one of the most commonly used forms of the mutation operator in DE (*DE/rand/1*), three different vectors are chosen from the population randomly and a mutant vector is created using through the equation below, where F is the mutation factor. This mutant vector is called the *donor vector*.

$$V_{i,g} = X_{r0,g} + F(X_{r1,g} - X_{r2,g})$$

where $V_{i,g}$ is the donor vector and $X_{ri,g}$ are the randomly chosen vectors.

As can be seen from the definition of the mutation operator of the DE algorithm, it is not possible to use it for binary valued problems without a modification. There are some approaches in literature for modifying DE for such binary valued problems. One of these methods uses an *angle modulation technique* to transform the binary space into a continuous space [29]. In the initial testing stage of this study, the experimental results obtained using this technique turned out to be insufficient. So the approach proposed in [30] is used as the binary DE implementation in this study. The details of this method are explained further in the following paragraphs.

```
randomly create initial population;
evaluate population;
repeat
      for population_size times do
         select next target vector;
         randoly choose base vector;
         donor vector = mutate (base vector);
         trial vector = crossover (target vector, donor vector);
         if trial vector better than target vector then
               select trial vetor;
         else
               select target vector;
         endif
      endfor
until max_fitness_evaluations reached;
```

Figure 5. Basic Differential Evolution Algorithm.

The initialization step randomly sets the initial values of the parameters in the population, to be either 1 or 0. The modification on DE to make it run within binary spaces is done to the mutation operator.

According to the approach proposed in [30], the multiplication, addition and substitution operations are achieved using the Hamming distance [31] between the two vectors. After the substitution step, each parameter in the vector is multiplied with the F parameter. This operation forces the values of the parameters to change from the binary space into the continuous space. In the next step of the mutation operator, which is addition, the values are transformed back into being either 0 or 1 through a rounding mechanism. In the crossover step, part of the new vector is taken from the target vector while the rest is taken from the donor vector through a mechanism similar to two-point crossover in GAs using a Cr parameter as the probability in determining the length of the segment taken from the target vector. The vector created through crossover is called the *trial vector*. The beginning of the segment to obtain from the target vector is determined randomly. The length L of the segment is determined as given in Figure 6. For larger values of Cr, a larger portion of the new vector is taken from the target vector.

```
L=0;
 do
    L=L+1:
  while (random() < Cr) AND
(L<Chromosome_Length);
```

Figure 6. Determining the length of the segment.

In the selection step, either the target vector or the trial vector is chosen based on their fitness values. These steps continue until a predefined number of maximum DE iterations have been reached.

4. Experiments

The results of the EAs are compared based on three datasets. Two of these are commonly used benchmark datasets found in literature. The last is real-world data obtained from the Turkish Interconnected Power System. This data is extracted from documents collected from 8 generators in Turkey in 2006 and is processed to convert into UCP formulation.

In [13], results obtained through other approaches for the two benchmark datasets are reported. These results are also used in this paper for comparisons. These approaches are:

- GGA: a standard generational GA, as used in [13];
- MA: a Memetic Algorithm [32], as used in [13], where the standard GGA is enhanced with a modified Lamarckian approach with local search.

4.1. Experimental Setup

The following parameter settings are used in the EAs in the experiments. All parameter values are determined using the best settings found as a result of a series of experimental runs.

In all EAs the population size is taken as 100. This corresponds to $\lambda = 100$ in ES and $\mu = 700$ is chosen for the size of the child population. Population size is important. When it is chosen very low, there will not be sufficient diversity in the population and the search may get stuck at local optima. However, very large population sizes mean too many fitness evaluations. Since fitness evaluations are the most time consuming

steps of the problem (due to the fact that lambda-iteration has to be run in each fitness evaluation), the running times may increase drastically. Initial parameter tuning tests were performed for all approaches to determine 100 as a common population size for which each approach performed well within acceptable times.

All EAs are allowed to do a maximum of 5000 fitness evaluations for the first and third datasets and 20000 evaluations for the second dataset. Initial σ_i values for ES are set as $1/length$ where $length$ is the length of the chromosome. The crossover probability in SSGA and ES is taken as $pc = 1$, which means that all pairs go through crossover. In the DE, C_r is taken as 0.1 and F is taken as 0.6. In lambda iteration, the tolerance is set to 0.0001. 20 runs of each of the algorithms with different random seeds are performed and the best, average and worst feasible results obtained for the total costs are reported here.

An individual in the EA corresponds to a potential solution to the UCP. For a system with N units and a time horizon of T hours, the chromosome length is $N \times T$, where each gene shows whether the corresponding generator is on or off for that time slot. The fitness of an individual depends on the cost of the solution S as well as the penalty values it acquires:

$$fitness(S) = cost(S) + penalty(S)$$

Cost for power generation is calculated using the lambda-iteration method based on the status of each power generator unit. For each hour, depending on whether the start-up is a cold start or a hot start, the appropriate cost is added to the total cost:

$$cost(S) = fuelCost(S) + startupCost(S).$$

Both the fuel cost and the start-up cost values are calculated as explained in Section 2. The penalty value is composed of two parts:

$$penalty(S) = M * demandPenalty(S) + K * updownPenalty(S)$$

where M and K are constant coefficients. Calculation of the demand penalty and up/down penalty terms is taken from [13]. A penalty term is used if the hourly power demands plus a specific amount of reserve is not met or if t_{up} and t_{down} constraints are violated. The multiplier M for the penalty term representing the amount of unmet power demand is set to 200. The multiplier K for the second penalty term representing the violated up/down constraint is taken as 10. Details on the fitness evaluation and the penalty calculation method can be found in [13].

4.2. Results

The first test problem [33] has 4 power generating units and a time horizon of 8 hours. The data for this test system is given in Table 7 and Table 8. For the second test, a larger dataset [13] consisting of 10 generating units and a time horizon of 24 hours is used. The data and the results for this test are given in Table 9 and Table 10. For the third test, the real-world data from the Turkish Interconnected Power System is used. There are 8 generating units and a time horizon of 8 hours. The data for this test system is given in Table 11 and Table 12.

The results of the three EAs and the results reported in literature are given in Table 2, Table 3 and Table 4 for the three tests respectively. In the columns of the tables the following notation is used:

- Best: best result found over 20 runs;
- Avg: average of the results found over 20 runs;

- Worst: worst result found over 20 runs.

In the rows of the tables, the different approaches are abbreviated as such:

- SSGA: Steady-State Genetic Algorithm ;
- ES: Evolutionary Strategies;
- DE: Differential Evolution;
- GGA: Generational Genetic Algorithm (results taken from [13]);
- MA: Memetic Algorithm (results taken from [13]).

For the test results of the Turkish Interconnected Power System, solutions from the other EAs are not available since the real-world data used is obtained locally. Therefore only the results of the three EAs used in this paper are reported in Table 4.

Table 2. Results for Test System 1.

	Best	Avg	Worst
SSGA	74675	74675	74675
DE	74675	74784	75008
MA	74675	74909	75012
ES	74675	74966	75008
GGA	74675	n/a	n/a

Table 3. Results for Test System 2.

	Best	Avg	Worst
DE	565827	565965	566650
MA	565827	566453	566861
ES	565827	569199	571312
GGA	565866	567329	571336
SSGA	566564	569097	571532

Table 4. Results for the Turkish Interconnected Power System (System 3).

	Best	Avg	Worst
DE	530346	530346	530346
ES	530392	530392	530392
SSGA	530392	530392	530392

As can be seen, for the first test, which is smaller in problem size and thus easier than the other two, all EAs give the same best result. This result is reported as being the global optimum for this problem in [13]. However when we look at average and worst case results, we see that SSGA is better than the others because it locates the global optimum in all the test runs while the MA, ES and DE produce solutions distributed in a wider range.

For the second test, DE, MA and ES all locate the same best solution. When we look at the average and worst performances, DE seems to be better than all the others. MA is the second best performer but

not quite as good as DE. ES, GGA and SSGA are worse than the first two with GGA and SSGA locating a solution of lower quality than even ES, which is closer to them in average and worse case performances. The performance of MA is the closest to the best performer DE, however it should be kept in mind that while the EAs used in this paper are in their most basic forms and no special operators are implemented, the MA [13] uses specialized operators and an extra local search step through hill-climbing during the iterations, which is very time consuming. Therefore DE performs fewer actions to find better results than those of a state-of-the-art algorithm from literature.

For the third test, which again is smaller and thus seems to be easier than the second, DE is the best performer, locating the same good solution in every run while ES and SSGA find a slightly worse solution in all their runs. However the differences in solution quality are not too high.

As expected, the SSGA performs similarly to GGA since they use the same operators and the same application order, with the only difference being in the population replacement techniques used. The DE results in this paper are much improved than in the previous study [26] by the authors. In that study, only best results found over 20 runs were reported. Therefore in Table 5 we only list the best results in the previous and current studies for the DE on the same datasets and in Table 6 we give the different parameter settings which produced the results given in Table 5. The parameters of DE which are not listed in Table 6 have the same settings in both studies so they have not been shown in the table.

Table 5. Results of two studies using DE.

	Previous [26]	Current
Test 1	74675	74675
Test 2	566166	565827
Test 3	532142	530346

Table 6. Parameter settings in the two studies using DE.

	Previous [26]	Current
Cr	0.3	0.1
F	0.8	0.6
M	200	200
K	50	10

The improvement in the results is quite high for the second and third tests. This improvement is a result of extensive experimentation with the F, Cr, M and K parameters to find the best settings. This issue points to one shortcoming of the DE implementation used in this study: that performance is highly dependent on appropriate parameter settings.

5. Conclusion

The use of EAs for the unit commitment problem (UCP) is explored in this study. Three sets of tests are performed using a SSGA, an ES and a DE as the appropriate EAs. The first two tests are on benchmark datasets obtained from the literature. The results of these two tests are compared to those of current state-of-the-art evolutionary algorithms, namely a MA and a GGA found in literature. Then, the three EAs are further applied to the real-word data obtained from the Turkish Interconnected Power System.

As can be seen from the results obtained for the benchmark tests, the DE implementation is the best performer and the results of the other two EAs used in this study are comparable to those of the existing

approaches. As also stated above, it should be noted that while the EAs used in this study are in their most basic forms, the MA use local search through hill climbing during execution, which is computationally very costly. Thus the EAs perform fewer actions to find good results. The similar performance of SSGA and GGA is to be expected since they share the same operators and application order with no extra operations as in the MA.

These experimental results show that EAs, especially DE, are very suitable for the UCP. In this study, the best settings for all the EAs are determined as a result of extensive experimental runs, however it is especially seen in the case of DE, that performance may be dependent on the selection of some of the parameters. This should be thoroughly explored, making modifications to make the algorithms less susceptible to parameter settings. Overall, the tested EAs perform well on the UCP and the results promote further study.

Acknowledgements

The authors would like to acknowledge the work of Ali Keleş. The binary differential evolution implementation in [26] belongs to Ali Keleş. This program has been run in this study to obtain the differential evolution results.

References

[1] P. G. Lowery, "Generating Unit Commitment by Dynamic Programming", IEEE Transactions on Power Apparatus and Systems Vol. PAS-85, No. 5, pp. 422-426, 1966.

[2] W. J. Hobbs, G. Hermon, S. Warner, G. B. Sheble, "An Enhanced Dynamic Programming Approach for Unit Commitment", IEEE Transactions on Power Systems, Vol. 3, No. 3, pp. 1201-1205, 1988.

[3] C. Cheng, C. W. Liu, and C. C. Liu, "Unit Commitment by Lagrangian Relaxation and Genetic Algorithms", IEEE Transactions on Power Systems, Vol. 15, No. 2, pp. 707-714, 2000.

[4] A. Borghetti. A. Frangioni, F. Lacalandrs, C. A. Nucci, "Lagrangian Heuristics based on Disaggregated Bundle Methods for Hydrothermal Unit Commitment", IEEE Transactions on Power Systems, Vol. 18, No. 2, pp. 974, 2003.

[5] Z. Liu, N. Li, C. Zhang, "Unit Commitment Scheduling Using a Hybrid ANN and Lagrangian Relaxation Method", in Proceedings of International Conference on Multimedia and Ubiquitous Engineering, pp. 481-484, 2008.

[6] K. Tokoro, Y. Masuda, H. Nishini, "An Efficient Unit Commitment Schedule by Combining of Continuous Relaxation Method and Genetic Algorithm", IEEJ Transactions on Electronics, Information and Systems, Vol. 127. No. 9, pp. 1452-1459, 2007.

[7] C. L. Chen, S. C. Wang, "Branch-and-Bound Scheduling for Thermal Generating Units", IEEE Transactions on Energy Conversion, Vol. 8, No. 2, pp. 184-189, 1993.

[8] G. Cote, M. A. Laughton, "Decomposition Techniques in Power System Planning: the Benders Partitioning Method", Electrical Power and Energy Systems, Vol. 1, No. 1, pp. 57-64, 1979.

[9] F. Zhuang, F. D. Galiana, "Unit Commitment by Simulated Annealing", IEEE Transactions on Power Systems, Vol. 5, No. 1, pp. 311-317, 1990.

[10] A. H. Mantawy, Y. L. Abdel-Magid, S. Z. Selim, "Unit Commitment by Tabu Search", IEE Proceedings - Generation, Transmission and Distribution, Vol. 145, No. 1, pp. 56-64, 1998.

[11] S. A. Kazarlis, A. G. Bakirtzis, V. Petridis, "A Genetic Algorithm Solution to the Unit Commitment Problem", IEEE Transactions on Power Systems, Vol. 11, No. 1, pp. 83-92, 1996.

[12] A. Rudolf, R. Bayrleithner, "A Genetic Algorithm for Solving the Unit Commitment Problem of a Hydro-Thermal Power System", IEEE Transactions on Power Systems, Vol. 14, No. 4, pp. 1460-1468, 1999.

[13] J. Valenzuela, A. E. Smith, "A Seeded Memetic Algorithm for Large Unit Commitment Problems", Journal of Heuristics, Vol. 8, pp. 173-195, 2002.

[14] G. Dudek, "Unit Commitment by Genetic Algorithm with Specialized Search Operators", Electric Power Syst. Res., Vol.72, No. 3, pp. 299-308, Elsevier, 2004.

[15] J. Maturana, M. C. Riff, "Solving the Short-Term Electrical Generation Scheduling Problem by an Adaptive Evolutionary Approach", European J. of Operational Research, Vol. 179, pp. 677-691, 2007.

[16] Y. M. Chen, W. S. Wang, "Fast Solution Technique for Unit Commitment by Particle Swarm Optimisation and Genetic Algorithm", International Journal of Energy Technology and Policy, Vol. 5, No. 4, pp. 440-456, 2007.

[17] S. Patra, S. K. Goswami, B. Goswami, "Differential Evolution Algorithm for Solving Unit Commitment with Ramp Constraints", Electric Power Components and Systems, Vol. 36, No. 8, pp. 771-787, 2008.

[18] N. P. Padhy, "Unit Commitment -A Bibliographical Survey", IEEE Transactions on Power Systems, Vol. 19, No. 2, pp. 1196-2005, 2004.

[19] S. Salam, "Unit Commitment Solution Methods", Proceedings of World Academy of Science, Engineering and Technology, Vol. 26, pp. 600-605, 2007.

[20] I. J. Raglend, N. P. Padhy, "Comparison of Practical Unit Commitment Solutions", Electric Power Components and Systems, Vol. 36, No. 8, pp. 844-863, 2008.

[21] A. E. Eiben, J. E. Smith, *Introduction to Evolutionary Computing*, Springer Verlag, 2003.

[22] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[23] I. Rechenberg, *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Fomman Holzboog Verlag, 1973.

[24] L. J. Fogel, A. J. Owens, M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, New York, 1966.

[25] K. V. Price, R. M. Storn, J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, 2005.

[26] A. Keleş, A. Ş. Etaner-Uyar, B. Türkay, "A Differential Evolution Approach for the Unit Commitment Problem", ELECO 2007: 5th International Conference on Electrical and Electronics Engineering, Bursa, pp. 132-136, 2007.

[27] S. Damousis, A. G. Bakirtzis, "Genetic Algorithm Solution to the Economic Dispatch Problem", IEE Proceedings-C, Vol. 141, No. 4, pp. 377-382, 1996.

[28] H. G. Beyer, H. P. Schwefel, "Evolution Strategies, A Comprehensive Introduction", Natural Computing, Vol. 1, pp. 3-52, Kluwer, 2002.

[29] G. Pampara, A. P. Engelbrecht, N. Franken, "Binary Differential Evolution", IEEE Congress on Evolutionary Computation, pp. 1873-1879, 2006.

[30] T. Gong, A. Tuson, "Differential Evolution for Binary Encoding", 11th Online World Conference on Soft Computing in Industrial Applications, 2006.

[31] R. W. Hamming, "Error-detecting and Error-correcting Codes", Bell Systems Technical Journal, Vol. 29, pp. 147-160, 1950.

[32] N. Krasnogor, J. Smith, "A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues", IEEE Transactions on Evolutionary Computation, Vol. 9, No. 5, pp. 474-488, 2005.

[33] A. J. Wood, B. F. Wollenberg, *Power Generation, Operation, and Control*, Wiley, New York, 1996.

Appendix. System Data used in the Experiments.

Table 7. Test System 1 [33].

	Unit 1	Unit 2	Unit 3	Unit 4
Pmax (MW)	300	250	80	60
Pmin (MW)	75	60	25	20
a0	684.74	585.62	213.0	252.0
a1	16.83	16.95	20.74	23.60
a2	0.0021	0.0042	0.0018	0.0034
tup (h)	5	5	4	1
tdown (h)	4	3	2	1
Shot ($)	500	170	150	0
Scold ($)	1100	400	350	0.02
tcoldstart (h)	5	5	4	0
Initial State(h)	8	8	-5	-6

Table 8. Demand and Reserve Loads for Test System 1 [33].

Hour	**1**	**2**	**3**	**4**
Demand (Unit?)	450	530	600	540
Reserve (Unit?)	45	53	60	54
Hour	**5**	**6**	**7**	**8**
Demand (Unit?)	400	280	290	500
Reserve (Unit?)	40	28	29	50

Table 9. Test System 2 [13].

	Unit 1	**Unit 2**	**Unit 3**	**Unit 4**
Pmax (MW)	455	455	130	130
Pmin (MW)	150	150	20	20
a0	1000	970	700	680
a1	16.19	17.26	16.60	16.50
a2	0.00048	0.00031	0.00200	0.00211
tup (h)	8	8	5	5
tdown (h)	8	8	5	5
Shot ($)	4500	5000	550	560
Scold ($)	9000	10000	1100	1120
tcoldstart (h)	5	5	4	4
Initial State (h)	8	8	-5	-5
	Unit 5	**Unit 6**	**Unit 7**	**Unit 8**
Pmax (MW)	162	80	85	55
Pmin (MW)	25	20	25	10
a0	450	370	480	660
a1	19.70	22.26	27.74	25.92
a2	0.00398	0.00712	0.00079	0.00413
tup (h)	6	3	3	1
tdown (h)	6	3	3	1
Shot ($)	900	170	260	30
Scold ($)	1800	340	520	60
tcoldstart (h)	4	340	520	60
Initial State (h)	-6	-3	-3	-1
	Unit 9	**Unit 10**		
Pmax (MW)	55	55		
Pmin (MW)	10	10		
a0	665	670		
a1	27.27	27.79		
a2	0.00222	0.00173		
tup (h)	1	1		
tdown (h)	1	1		
Shot ($)	30	30		
Scold ($)	60	60		
tcoldstart (h)	0	0		
Initial State (h)	-1	-1		

Table 10. Demand and Reserve Loads for Test System 2 [13].

Hour	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Demand (Unit?)	750	750	850	950	1000	1100	1150	1200
Reserve (Unit?)	75	75	85	95	100	110	115	120
Hour	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Demand (Unit?)	1300	1400	1450	1500	1400	1300	1200	1050
Reserve (Unit?)	130	140	145	150	140	130	120	105
Hour	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Demand (Unit?)	1000	1100	1200	1400	1300	1100	900	800
Reserve (Unit?)	100	110	120	140	130	110	90	80

Table 11. Turkish Interconnected Power System.

	Unit 1	**Unit 2**	**Unit 3**	**Unit 4**
Pmax (MW)	1120	1350	1432	600
Pmin (MW)	190	245	318	150
a0	6595.5	7290.6	6780.5	1564.4
a1	7.0063	7.2592	5.682	3.1288
a2	0.0168	0.0127	0.0106	0.0139
tup (h)	8	1	1	10
tdown (h)	2	0,5	0,5	3
Shot ($)	800	800	600	400
Scold ($)	1600	1600	1200	800
tcoldstart (h)	8	1	1	10
Initial State (h)	-4	-4	-4	-4
	Unit 5	**Unit 6**	**Unit 7**	**Unit 8**
Pmax (MW)	990	420	630	630
Pmin (MW)	210	110	140	140
a0	5134.1	1159.5	1697	1822.8
a1	6.232	3.3128	3.2324	3.472
a2	0.0168	0.021	0.013	0.0147
tup (h)	10	10	10	10
tdown (h)	3	3	3	3
Shot ($)	500	400	400	400
Scold ($)	1000	800	800	800
tcoldstart (h)	10	10	10	10
Initial State (h)	-4	-4	-4	-4

Table 12. Demand and Reserve Loads for the Turkish Interconnected Power System.

Hour	**1**	**2**	**3**	**4**
Demand (Unit?)	2000	3000	6500	1500
Reserve (Unit?)	200	300	650	150
Hour	**5**	**6**	**7**	**8**
Demand (Unit?)	4200	5100	2700	1750
Reserve (Unit?)	420	510	270	175