Adaptive general variable neighborhood search heuristics for solving the unit commitment problem

Raca Todosijević, Marko Mladenovic, Said Hanafi, Nenad Mladenovic, Igor Crevits

Electrical Power and Energy Systems 78 (2016) 873–883

Adaptive general variable neighborhood search heuristics for solving the unit commitment problem

Raca Todosijević [a,b,*], Marko Mladenović [a], Saïd Hanafi [a], Nenad Mladenović [a,b], Igor Crévits [a]

[a] LAMIH UMR CNRS 8201 – Université de Valenciennes, 59313 Valenciennes Cedex 9, France
[b] Mathematical Institute, Serbian Academy of Science and Arts, 11000 Belgrade, Serbia

article info

Keywords:
Power systems
Unit commitment problem
Mixed integer nonlinear problem
Variable neighborhood search

abstract

The unit commitment problem (UCP) for thermal units consists of finding an optimal electricity production plan for a given time horizon. In this paper we propose hybrid approaches which combine Variable Neighborhood Search (VNS) metaheuristic and mathematical programming to solve this NP-hard problem. Four new VNS based methods, including one with adaptive choice of neighborhood order used within deterministic exploration of neighborhoods, are proposed. A convex economic dispatch subproblem is solved by *Lambda iteration* method in each time period. Extensive computational experiments are performed on well-known test instances from the literature as well as on new large instances generated by us. It appears that the proposed heuristics successfully solve both small and large scale problems. Moreover, they outperform other well-known heuristics that can be considered as the state-of-the-art approaches.

Introduction

The unit commitment problem (UCP) consists of determining optimal production plan for a given set of power plants over a given time horizon so the total production cost is minimized, while satisfying various constraints. Every power plant individually needs to satisfy: minimum up time (minimal number of consecutive time periods during which the unit must be turned on), minimum down time (minimal number of consecutive time periods during which unit must be turned off) and production limit constraints (lower and upper production bounds). The total production of all active plants must satisfy the required demand minding that the maximal possible production cannot be less than the sum of required demand and required spinning reserves.

The unit commitment problem can be formulated as a mixed integer nonlinear problem (MINLP). Binary variables represent the ON/OFF state of every unit for each time period, while continuous variables quantify the unit production expressed in megawatts for each time period. It is easy to conclude that the number of all possible solutions grows exponentially by increasing the number of plants. The UCP is NP-hard, which means that it cannot be exactly solved in reasonable amount of time. This holds even for moderate number of units, therefore, many heuristics have been proposed in the literature to solve the UCP approximatively.

The exact method based on dynamic programming [13,14,27,35] for solving the UCP was able to tackle only problems with small number of units. Many heuristic and metaheuristic methods have been proposed up to now for the UCP such as: priority list method [1], genetic algorithms [5,22,48,53], tabu search algorithms [38], particle swarm optimization algorithms [46,56], ant colony algorithms [42], fuzzy logic [11], artificial neural networks [9,47], evolutionary programming [21], simulated annealing [43–45]. For other UCP related problems and solution approaches, we refer the reader to [39,57] and references therein.

In this paper we propose hybrid approaches that combine Variable Neighborhood Search (VNS) metaheuristic with mathematical programming. We in fact substantially extend our conference paper [54] by considering an adaptive VNS approach. Four new VNS based methods, including one with adaptive choice of neighborhood order used within deterministic exploration of neighborhoods, are proposed. Benchmark instances were used to test our hybrid methods. They have been compared with other heuristics proposed in the literature. Moreover, we suggest new set of large size instances. Computational results show that the proposed heuristic outperforms all current heuristic approaches, while improving running times for most instances. It is especially true for the largest size instances.

The rest of the paper is organized as follows. In Section 'Problem formulation' we provide mathematical formulation of the UCP, in Section 'General variable neighborhood search for solving the UCP' we describe our method. Section 'Computational results' presents comparison of our method with existing approaches, while

E-mail addresses: racatodosijevic@gmail.com (R. Todosijević), mladja87@gmail.com (M. Mladenović), said.hanafi@univ-valenciennes.fr (S. Hanafi), nenad.mladenovic@univ-valenciennes.fr (N. Mladenović), igor.crevits@univ-valenciennes.fr (I. Crévits).

DOI : 10.1016/j.ijepes.2015.12.031

Section 'Concluding remarks' concludes the paper and offers directions for future research.

Problem formulation

Economic dispatch problem

Before introducing the UCP it is necessary to define its subproblem: the Economic Dispatch Problem (EDP). Consider n thermal units (power generation units fueled by coal, oil or gas) committed to serve a load of P^D, at minimum cost. Every (thermal) unit production is bounded from below and above, this means that each unit has minimal and maximal production capacities. The objective of the EDP is to minimize the production cost while satisfying the required load and generation limit constraints for each unit. This problem can be formulated as follows:

$$\min \quad F = \sum_{i=1}^{n} F_i(P_i)$$

subject to

$$\sum_{i=1}^{n} P_i = P^D \tag{1}$$

$$P_i^{min} \leqslant P_i \leqslant P_i^{max}, \quad \text{for } i = 1, \ldots, n \tag{2}$$

where

- P_i is the production of unit i (in MW),
- $F_i(P_i)$ is the cost of production by unit i (in \$/h),
- P^D is the demand (in MW).

Fuel cost function of each unit is set as a quadratic function [58]

$$F_i(P_i) = a_i + b_i P_i + c_i P_i^2, \tag{3}$$

where $a_i, b_i, c_i, \quad i = 1, \ldots, n$ are given coefficients.

Lambda iteration method for solving economic dispatch problem

The lambda-iteration method [58] is, so far, the most popular method for solving the EDP, when the objective function F is quadratic. It is used to iteratively determine optimal Lagrange multiplier λ which corresponds to constraint (1). The lambda iteration procedure stops when the tolerance, which indicates that the sum of all online units output minus the load demand, is less than the value given beforehand. When the Lagrange multiplier λ is known, it is simple to calculate the production of each unit by solving system of linear equations. The scheme of lambda iteration method is presented below (Algorithm 1).

Algorithm 1. Lambda Iteration method

Function LIM();
$\lambda^{min} \leftarrow \min_{i=1,\ldots,n} \frac{dF_i(P_i^{min})}{dP_i}$;
$\lambda^{max} \leftarrow \max_{i=1,\ldots,n} \frac{dF_i(P_i^{max})}{dP_i}$;
$\epsilon \leftarrow 10^{-6}$;
repeat
  $\lambda \leftarrow (\lambda^{min} + \lambda^{max})/2$;
  Calculate P_i from $\frac{dF_i(P_i)}{dP_i} = \lambda$;
  $\Delta = P^D - \sum_{i=1}^{n} P_i$;
  if $P^D > \sum_{i=1}^{n} P_i$ **then** $\lambda^{min} = \lambda$;
  if $P^D < \sum_{i=1}^{n} P_i$ **then** $\lambda^{max} = \lambda$;
until $|\Delta| \leqslant \epsilon$;
return $P_1, \ldots, P_n$;

Unit commitment problem

The basic goal of the UCP is to properly schedule the ON/OFF states of all units in the system with minimum (fossil) fuel cost. The ON/OFF state of the entire system is represented by the binary matrix $U_{i,t} \in \{0, 1\}$, for $i \in N = \{1, \ldots, n\}$ and $t \in H = \{1, \ldots, T\}$. In addition to fulfill a large number of constraints, the optimal UC should meet the predicted load demand requirement (7) with spinning reserves (6) at every time interval such that the total operating cost is minimal (4). Therefore, the solution of the unit commitment problem relies on iteratively solving the economic dispatch problem for all time intervals t (e.g., an hour) in overall time T respecting feasibility of the time constraints (8).

The model can be stated as follows:

$$\min \quad G = \sum_{i \in N} \sum_{t \in H} \left[\left[F_i(P_{i,t}) + ST_i(1 - U_{i,t-1}) \right] U_{i,t} + SD_i U_{i,t-1}(1 - U_{i,t}) \right] \tag{4}$$

subject to:

$$P_i^{min} U_{i,t} \leqslant P_{i,t} \leqslant P_i^{max} U_{i,t}, \quad i \in N, \ t \in H \tag{5}$$

$$\sum_{i \in N} P_i^{max} U_{i,t} \geqslant P_t^D + P_t^R, \quad t \in H \tag{6}$$

$$\sum_{i \in N} P_{it} = P_t^D, \quad t \in H \tag{7}$$

$$\begin{aligned} &U_{i,t} - U_{i,t-1} \leqslant U_{i,t+j} \quad i \in N;\ t \in H;\ j = 1, \ldots, T_{i,up} - 1; \\ &U_{i,t+j} \leqslant U_{i,t} - U_{i,t-1} + 1 \quad i \in N;\ t \in H;\ j = 1, \ldots, T_{i,down} - 1; \end{aligned} \tag{8}$$

$$U_{i,t} \in \{0, 1\}, \quad P_{i,t} \geqslant 0 \quad i \in N;\ t \in H \tag{9}$$

where

$$ST_i = \begin{cases} HSC_i, & \text{if } T_{i,down} \leqslant T_{i,off}^t \leqslant T_{i,cold} + T_{i,down} \\ CSC_i, & \text{otherwise} \end{cases} \tag{10}$$

It is assumed that the shut down cost SD_i for every unit i is equal to zero ($SD_i = 0$).

The startup cost (ST_i) depends on how long unit i is off line. We differ two types of startup cost: cold start (CSC_i) and hot start (HSC_i). In practice, the cold start is much more expensive than a hot start.

The constraints (8) represent the minimum up and down time requirements of each unit. This means that each unit must be on line (up time) and off line (down time) for a certain consecutive time period ($T_{i,up}$, $T_{i,down}$, respectively). These two constraints represent the time constraints, while (5)–(7) are production constraints. This means that a UCP solution must fulfill both production feasibility (produce required load, bounded by system capacities, satisfying spinning reserve constraints) and time feasibility (units must be online/offline for a consecutive time period).

The presented formulation of the UCP has $nT + 2T + nT\sum_{i=1}^{n}(T_{i,up} + T_{i,down} - 2)$ constraints and $O(nT)$ binary and continuous variables.

General variable neighborhood search for solving the UCP

As we mentioned earlier, finding an optimal solution for large size the UCP is unlikely to be possible in reasonable time, and thus heuristic methods are a preferable option for finding good or near-optimal solutions. For that purpose, we propose an efficient Variable Neighborhood Search (VNS) based heuristics [16,31].

VNS is a flexible framework for building heuristics to approximately solve combinatorial and non-linear continuous optimization problems. VNS changes systematically the neighborhood structures during the search for an optimal (or near-optimal)

solution. The changing of neighborhood structures is based on the following observations: (i) A local optimum relatively to one neighborhood structure is not necessarily a local optimum for another neighborhood structure; (ii) A global optimum is a local optimum with respect to all neighborhood structures; and (iii) Empirical evidence shows that for many problems all local optima are relatively close to each other. The first property is exploited by increasingly using complex moves in order to find local optima with respect to all neighborhood structures used. The second property suggests using several neighborhoods, if local optima found are of poor quality. Finally, the third property suggests exploitation of the vicinity of the current incumbent solution.

Neighborhood structures

If the state of each unit in each time period is given, so that constraints (6)–(9) are fulfilled, then the problem (4)–(9) becomes an economic dispatch problem (EDP), and the corresponding optimal production plan, in each time period, may be obtained by solving the appropriate EDP. In order to take the advantage of this fact and the fact that economic dispatch can be solved efficiently (see Section 'Lambda iteration method for solving economic dispatch problem'), we propose to present a feasible solution of the UCP with a $n \times T$ matrix U whose entries $U_{i,t}$ satisfy constraints (6)–(9). Total operating cost of the feasible solution U is denoted by $G(U)$. Next, we propose neighborhood structures to efficiently explore the solution space of matrices U (that satisfy constraints (6)–(9)). Let us define the set of all solutions with $N'_k(U)$, obtained by changing exactly k values of the matrix U. Obviously, such a set contains not only feasible solutions, but also solutions that violate some constraints. This issue is resolved by using procedure proposed in [6]. Let us denote with M the set of all solutions which can be obtained by repairing infeasible solutions from $N'_k(U)$. Now, we define the k-th neighborhood of solution U ($N_k(U)$) as the union of $N''_k(U)$ and M, where $N''_k(U)$ represents the set of all feasible solutions from $N'_k(U)$. As already mentioned, the production cost of each solution $U' \in N_k(U)$ is determined by solving the series of economic dispatch sub-problems.

Generating an initial solution

Priority list

The merit order is obtained based on the average fuel cost of unit operating at certain fixed fraction of maximum output. The merit order of unit j is defined as

$$M_j = \frac{2 \cdot F((P_{j,max} + P_{j,min})/2)}{P_{j,max} + P_{j,min}} \tag{11}$$

The previously defined merit order is used for building the so-called Priority list (PL) which contains units sorted according to increasing merit order.

Generating greedy solution

For generating greedy solution we use procedure proposed in [6], which steps are described in Algorithm 5.

Firstly, the procedure builds a solution, that satisfies power balance constraints and spinning reserve constraints, committing units according to the Priority list (Algorithm 2). The solution obtained this way will most probably be infeasible, because the used procedure neglects minimum up and down constraints. However, the feasibility of this solution can be changed by repairing minimum up and down time using a heuristic procedure described below (Algorithm 3). It should be noted that procedure repairs minimum up and down time without violating other constraints.

To check for violations of minimum up and down constraints, the on and off states of units are determined in advance. The on/off states at hour t are calculated using the following formulas:

$$T^t_{i,on} = \begin{cases} T^{t-1}_{i,on} + 1 & \text{if } U_{i,t} = 1 \\ 0 & \text{otherwise} \end{cases}$$
$$T^t_{i,off} = \begin{cases} T^{t-1}_{i,off} + 1 & \text{if } U_{i,t} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Repairing the minimum up and down time constraints can lead to excessive spinning reserves, which is not desirable due to the high operation cost. For this reason, we use a heuristic search algorithm (Algorithm 4) based on the priority list to de-commit redundant units due to the minimum up and down time repairing, thereby reducing the operating cost. The algorithm searches for units that can be de-committed without violating constraints starting from the unit with highest values of M_i until there is no unit that can be de-committed.

Algorithm 2. Primary unit scheduling

Function `Primary`();
Calculate values M_i according to (11)
Sort units in ascending order of M_i
for $t = 1$ **to** T **do**
  For each unit i set $U_{i,t}$ to zero;
  while $\sum_{i=1}^{N} P_{i,max} U_{i,t} < P_{Dt} + P_{Rt}$ **do**
    Choose not-committed unit k with lowest value M_k;
    $U_{k,t} \leftarrow 1$;
  end
end
return U;

Algorithm 3. Repairing minimum up and down time

Function `Repair`(U);
Calculate continuous on and off times of all units using (12);
for $t = 1$ **to** T **do**
  for $i = 1$ **to** N **do**
    if ($U_{i,t} = 0$ `and` $U_{i,t-1} = 1$ `and` $T^{t-1}_{i,on} < T_{i,up}$) **then**
      $U_{i,t} = 1$;
    if ($U_{i,t} = 0$ `and` $U_{i,t-1} = 1$ `and` $T^{t+T_{i,down}-1}_{i,off} < T_{i,down}$) **then**
      $U_{i,t} = 1$;
    Update the on/off status for the unit i in (12);
  end
end
return U;

Algorithm 4. Deccomitment of excessive units

Function `Deccomit`(U);
for $t = 1$ **to** T **do**
  $E = \varnothing$;
  for $i = 1$ **to** N **do**
    if (unit i can be de-committed without violations) **then**
      $E = E \cup \{i\}$;
  end
  repeat
    Choose $k \in E$ with the highest value M_k;
    if (unit k can be de-committed without violating spinning reserves constraint) **then**
      $U_{k,t} \leftarrow 0$;
    $E = E \setminus \{k\}$;
  until $E \neq \varnothing$;
end
return U;

Algorithm 5. Building greedy solution

```
Function Greedy();
1 U ← Primary();
2 U ← Repair(U);
3 U ← Decommit(U);
  return U;
```

Pipe variable neighborhood descents

Variable neighborhood descent (VND) is a deterministic variant of VNS in which neighborhoods are ordered in a sequence and used one after another until a local minimum with respect to all of them is reached. Usually the search returns to the first neighborhood in the sequence whenever improvement in any neighborhood structure is obtained. This VND variant is called sequential VND (seqVND). Another option is to continue search in the same neighborhood in the case of detected improvement. Such variant we call pipe VND (pipeVND) [30]. In this paper we develop two variants of pipeVND for solving the UCP that differ in the order of local searches during the optimization process. Namely, both pipeVNDs use the same set of two neighborhoods sequentially explored one after another. Additionally, each of them is iterated until there is no improvement in objective function value (Algorithms 8 and 9). The used local searches attempt to de-commit units, preserving feasibility. They are based on the priority list, i.e., the search for units which will be de-committed is organized according to the descending merit order. The difference between these two local searches is in the number of consecutive time periods (hours) attempted to de-commit a unit. The first local search, LS1, attempts to de-commit a unit i for a period of T_i^{down} hours (see Algorithm 6), while the second, LS2, attempts to de-commit each unit in one hour (see Algorithm 7). Note that within the proposed local searches the production plans are re-calculated, solving the corresponding economic dispatch sub-problems only for the time periods where a de-commitment of a unit occurs. Both local searches explore the solution space using the first improvement strategy. The `pipeVND1` applies LS1 and then LS2, while `pipeVND2` employs LS2 and then LS1.

Algorithm 6. Local search 1

```
Function LS1(U);
1 E = {i_1, i_2, ... i_N}   /* indices of units sorted according to
    descending merit order */;
2 U' ← U;
3 for k = 1 to N do
4   for t = 1 to T-T_i^down+1 do
      if Unit i_k can be de-committed during time period [t,t
        +T_i^down-1] keeping feasibility then
5       for h = t to t + T_i^down − 1 do
6         U'_{i_k,h} ← 0;
7         Solve the economic dispatch problem for the time
            period h;
        end
8       if G(U') < G(U) then U ← U';
9         else U' ← U;
      end
    end
  end
  return U;
```

Algorithm 7. Local search 2

```
Function LS1(U);
1 E = {i_1, i_2, ... i_N}   /* indices of units sorted according to
    descending merit order */;
2 U' ← U;
3 for k = 1 to N do
4   for t = 1 to T do
      if Unit i_k can be de-committed in hour t keeping
        feasibility
5       U'_{i_k,t} ← 0;
6       Solve the economic dispatch problem for the time
          period t;
7       if G(U') < G(U) then U ← U';
8         else U' ← U;
      end
    end
  end
  return U;
```

Algorithm 8. pipeVND1

```
Function pipeVND1(U);
  repeat
1   U ← LS1(U) ;
2   U ← LS2(U) ;
  until There is no improvement;
  return U;
```

Algorithm 9. pipeVND2

```
Function pipeVND2(U);
repeat
1   U ← LS2(U) ;
2   U ← LS1(U) ;
until There is no improvement
return U;
```

General variable neighborhood search algorithms

Up to now, many variants of VNS have been proposed in the literature (see e.g., [15,16,26,32] for recent surveys). However, the widely used variant is the so-called General VNS (GVNS). It uses some variant of VND as a local search within basic VNS scheme. Recently, many adaptive variants of VNS that dynamically change their ingredients during the solution process have been proposed (see e.g., [18,23,24,36,40,51]).

For solving unit commitment problem, we developed two variants of GVNS which use the same shaking procedure `Shake`(U,k) in the diversification step. The function `Shake`(U,k) at the output returns a random solution from the k-th neighborhood of a given solution U. More precisely, the shaking procedure firstly changes exactly k random entries of a given matrix U, and after that, if the resulting solution is infeasible, applies the procedure from [6] to restore feasibility. The solution generated in this way is returned at the output. However, two proposed variants differ in the way of performing intensification. The first one called `GVNS`, uses the `pipeVND1` as a local search while the second one, called `Adaptive_GVNS`, decides whether the `pipeVND1` or the `pipeVND2` will be applied in some iteration of GVNS, depending on their

success in previous solution process. In the first iteration `Adaptive_GVNS` uses `pipeVND1`, while in all other iterations the decision which pipe VND will be applied is made as follows. Initially, both `pipeVND1` and `pipeVND2` have the same merit value $w = 0.5$ assigned to them. After that at each iteration their merits are updated dynamically. Namely, the merit value of used pipeVND variant is increased or decreased for some value u (e.g., $u = 0.1$) depending on whether the currently best found solution is improved or not in that iteration. The currently used pipeVND will be replaced by another in the next iteration if its merit becomes negative. If replacement of a pipeVND occurs, its merit is reset to the initial value w. The outline of both `GVNS` and `Adaptive_GVNS` are given at Algorithms 10 and 11, respectively.

Each of the proposed GVNS based heuristics, at the input, requires two parameters. The first one denoted by t_{max} represents the maximal running time of a heuristic, while the second one, named k_{max}, represents the maximum number of iterations that can be executed within the shaking procedure.

Algorithm 10. GVNS for unit commitment problem

```
Function GVNS (U, k_max, t_max);
1 repeat
2   k ← 1;
3   repeat
4     U' ← Shake(U, k);   /* Shaking */
5     U'' ← pipeVND1(U');   /* Local search */
6     k ← k + 1;   /* Next neighborhood */
7     if G(U'') < G(U) then
8       U ← U''; k ← 1;   /* Make a move */
      end
9     t ← CpuTime()
    until k = k_max;
  until t > t_max;
```

Algorithm 11. Adaptive GVNS for unit commitment problem

```
Function Adaptive_GVNS (U, k_max, t_max);
 1 repeat
 2   order ← 1;
 3   merit1 ← w;
 4   merit2 ← w;
 5   k ← 1;
 6   repeat
 7     U' ← Shake(U, k);   /* Shaking */
 8     if order = 1 then U'' ← pipeVND1(U'); /* Local search */
 9     if order = 2 then U'' ← pipeVND2(U'); /* Local search */
10     if G(U'') < G(U) then
11       U ← U''; k ← 1;   /* Make a move */
12       if order = 1 then merit1 ← merit1 + u;
13       if order = 2 then merit2 ← merit2 + u;
       else
14       k ← k + 1;   /* Next neighborhood */
15       if order = 1 then merit1 ← merit1 − u;
16       if order = 2 then merit2 ← merit2 − u;
17       if merit1 < 0 then order ← 2; merit1 ← w;   /*
           update order*/
18       if merit2 < 0 then order ← 1; merit2 ← w;   /*
           update order*/
       end
19     t ← CpuTime()
    until k = k_max;
  until t > t_max;
```

Computational results

Both previously described GVNS based heuristics, namely, `GVNS` and `Adaptive_GVNS`, are tested by constructing initial solutions in two different ways. If the initial solution is obtained by greedy Algorithm 5, the corresponding `GVNS` and `Adaptive_GVNS` variants are denoted by `GVNS-G` and `Adaptive_GVNS-G`, respectively. Similarly, if `GVNS` (`Adaptive_GVNS`) uses greedy randomized initial solution [12], that variants are named as `GVNS-R` (`Adaptive_GVNS-R`). The greedy randomized initial solutions for both GVNSs are built iteratively, starting from the greedy solution obtained by Algorithm 5. Each iteration consists of choosing a random solution from the first neighborhood of the current solution and setting the chosen solution to be the new current solution. The whole process is repeated $N \cdot T$ times.

In all experiments the value of the k_{max} parameter is set to 5, whereas the time limit is set to $t_{max} = 600$ s, for all developed heuristics unless stated otherwise.

Test instances

In order to perform empirical analysis, we use the following two data sets from the literature:

Case Study 1 [22]

This case study contains test instances with up to 100 units. The instances with more than 10 units are derived duplicating the data of the basic instance with 10 units. The load demands for those derived instances are adjusted in proportion to the number of units. The spinning reserve requirement, for all instances, is set to 10% of total load demand. The data for the basic 10 units test instance is provided in Appendix (Tables 9 and 10).

Case Study 2 [17]

The second case study consists of 38 generating units from the practical Taiwan Power (Taipower). The data of that system are provided in Appendix (Tables 11 and 12). The spinning reserve requirement is set to 11% of the total load demand.

Comparison of GVNS approaches with other heuristic approaches on Case Study 1 instances

The fuel costs obtained by our methods are compared with fuel costs obtained by the following 23 heuristics from the literature: Lagrangian relaxation (LR) [22]; genetic algorithm (GA) [22]; enhanced adaptive Lagrangian relaxation (ELR) [34]; Dynamic Programming with ELR (DPLR) [34]; Lagrangian relaxation and genetic algorithm (LRGA) [3]; genetic algorithm based on characteristic classification (GACC) [48]; evolutionary programming (EP) [21]; priority-list-based evolutionary algorithm (PLEA) [50]; extended priority list (EPL) [50]; integer coded genetic algorithm (ICGA) [5]; a Lagrangian multiplier based sensitive index to determine the unit commitment of thermal units (LMBSI) [49]; improved pre-prepared power demand and Muller method (IPPDTM) [2]; quantum inspired binary particle swarm optimization (QBPSO) [20]; quantum-inspired evolutionary algorithms (QEA-UC) [25] and (IQEA-UC) [4]; shuffled frog leaping algorithm (SFLA) [10]; imperialistic competition algorithm (ICA) [33]; gravitational search algorithm (GSA) [41]; semi-definite programming (SDP) [19,29]; tighter relaxation method (RM) [37]. Comparative results are given in Table 1. It should be emphasized that for the test instance with 20 units, LRGA, SFLA and GSA heuristics report solution values better than the optimal solution value (which equals to 1,123,297 see [55]). Therefore, we boldface that value in Table 1, but values better than optimal present in italic font. For instance with 40 units the

Table 1
Comparison of GVNS variants with 21 methods on Case Study 1 instances [22].

No. of units	10 TU's	20 TU's	40 TU's	60 TU's	80 TU's	100 TU's	Average
Method	Operating cost ($)						
LR [22]	565,825	1,130,660	2,258,503	3,394,066	4,526,022	5,657,277	2922058.83
ELR [34]	563,977	**1,123,297**	2,244,237	3,363,491	4,485,633	5,605,678	2897718.83
LRGA [3]	564,800	*1,122,622*	**2,242,178**	3,371,079	4,501,844	5,613,127	2902608.33
DPLR [34]	564,049	1,128,098	2,256,195	3,384,293	4,512,391	5,640,488	2914252.33
GA [22]	565,825	1,126,243	2,251,911	3,376,625	4,504,933	5,627,437	2908829.00
GACC [48]	563,977	1,125,516	2,249,715	3,375,065	4,505,614	5,626,514	2907733.50
EP [21]	564,551	1,125,494	2,249,093	3,371,611	4,498,479	5,623,885	2905518.83
ICGA [5]	566,404	1,127,244	2,254,123	3,378,108	4,498,943	5,630,838	2909276.67
PLEA [50]	563,977	1,124,295	2,243,913	3,363,892	4,487,354	5,607,904	2898555.83
EPL [50]	563,977	1,124,369	2,246,508	3,366,210	4,489,322	5,608,440	2899804.33
LMBSI [49]	563,977	1,123,990	2,243,708	3,362,918	4,483,593	5,602,844	2896838.33
IPPDTM [2]	563,977	–	2,247,162	3,366,874	4,490,208	5,609,782	–
QBPSO [20]	563,977	**1,123,297**	2,242,957	3,361,980	4,482,085	5,602,486	2896130.33
QEA-UC [25]	**563,938**	1,123,607	2,245,557	3,366,676	4,488,470	5,609,550	2899633.00
IQEA-UC [4]	**563,938**	**1,123,297**	2,242,980	3,362,010	4,482,826	5,602,387	2896239.67
SFLA [10]	564,769	*1,123,261*	2,246,005	3,368,257	4,503,928	5,624,526	2905124.33
ICA [33]	**563,938**	1,124,274	2,247,078	3,371,722	4,497,919	5,617,913	2903807.33
GSA [41]	**563,938**	*1,123,216*	2,242,741	3,362,447	4,483,864	5,600,883	2896181.50
SDP [19]	563,938	1,124,357	2,243,328	3,363,031	4,484,365	5,602,538	2896926.17
SDP [29]	563,977	1,124,410	2,243,144	3,360,512	4,480,652	5,598,727	2895237.00
RM [37]	563,977	1,123,990	2,243,676	3,361,589	4,481,833	5,599,761	2895804.33
GVNS-R	563,938	**1,123,297**	2,242,882	3,360,316	4,480,515	**5,597,962**	2894818.33
GVNS-G	563,938	**1,123,297**	2,242,882	3,360,699	4,480,617	5,600,133	2895261.00
Adaptive_GVNS-R	563,938	**1,123,297**	**2,242,596**	**3,360,181**	**4,480,328**	5,597,964	2894717.33
Adaptive_GVNS-G	563,938	**1,123,297**	2,242,882	3,361,119	4,480,617	5,598,876	2895121.50

Table 2
CPU time: Case Study 1 [22].

No. of units	10 TU's	20 TU's	40 TU's	60 TU's	80 TU's	100 TU's	Average
Method	CPU time (s)						
LMBSI [49]	10.00	18.00	27.00	40.00	54.00	73.00	37.00
IPPDTM [2]	0.52	–	6.49	17.39	31.23	46.55	20.44
QBPSO [20]	18.00	50.00	158.00	328.00	554.00	833.00	323.50
QEA-UC [25]	19.00	28.00	43.00	54.00	66.00	80.00	48.33
IQEA-UC [4]	34.00	98.00	146.00	191.00	235.00	293.00	166.17
ICA [33]	48.00	63.00	151.00	366.00	994.00	1376.00	499.67
GSA [41]	2.89	13.72	74.66	103.41	146.45	204.93	91.01
SDP [19]	25.41	63.94	157.73	260.76	353.84	392.56	209.04
RM [37]	1.15	2.14	4.83	8.79	13.02	17.10	7.84
GVNS-R	0.23	2.46	63.19	126.82	22.56	374.49	98.29
GVNS-G	0.05	2.5	6.64	436.35	98.76	351.56	149.31
Adaptive_GVNS-R	0.08	1.95	109.85	212.75	64.86	283.84	112.22
Adaptive_GVNS-G	0.04	1.23	2.14	109.53	287.49	552.49	158.82

optimal solution is not known. Hence, it is questionable if the value of 2,242,178 obtained by LRGA is really reliable (since it reported better value than optimal for the test instance with 20 units).

From results presented in Table 1 the following conclusion may be drawn:

- All proposed GVNS variants succeed in finding optimal solutions for instances with up to 20 units.
- For instances with 60 and 80 units, Adaptive_GVNS-R provides solutions of better quality than all proposed heuristics up to now in the literature. On the other hand, GVNS-R offers the best solution for instance with 100 units.
- Regarding the average solution cost achieved by each of compared heuristics, we conclude that Adaptive_GVNS-R outperforms all the others. The second best heuristic, turns out to be GVNS-R, while Adaptive_GVNS-G takes the third place in the overall ranking. GVNS-G is ranked as the fifth best immediately behind SDP heuristic [29].

The execution times of GVNS-R, GVNS-G, Adaptive_GVNS-R and Adaptive_GVNS-G as well as execution times of all other methods, are presented in Table 2. Note that the computer configurations for the methods of LMBSI [49], IPPDTM [2], QBPSO [20], QEA-UC [25], IQEA-UC [4], GSA [41], ICA [33], SDP [19], RM [37] are 2 GHz CPU, Pentium IV 2.8 GHz, Pentium IV 2.0 GHz, Intel Core 2.39 GHz, Intel core 2 Duo CPU 2.66 GHz, Intel Pentium IV 2-GHz CPU, Intel Core 2 Quad 2.4 GHz, core 2 duo processor 2 GHz, Intel Core 2 Duo Processor T5300 1.73 GHz and AMD Dual-Core 4800 + 2.5 GHz, respectively. All proposed GVNSs have been run on a computer with Intel i7 2.8 GHz CPU. All in all, all computer platforms have the similar characteristics. Note that our code is executed sequentially on a single core while some are executed on multiple CPU cores.

Comparison of GVNS approaches with exact methods on instances from Case Study 1

Results obtained by the proposed methods are compared with those obtained by Branch and Cut Search (B&C), SBB solver, DICOPT solver, CPLEX solver [28] and MILP-based approach [55]. The comparison is presented in Table 3.

Table 3
Comparison with exact methods: Case Study 1 [22].

Method	B&C	SBB	DICOPT	CPLEX	MILP	GVNS-R	GVNS-G	Adaptive_GVNS-R	Adaptive_GVNS-G
No. of units	Operating cost ($)								
10	**563,938**	572,468	**563,938**	564,189	**563,938**	**563,938**	**563,938**	**563,938**	**563,938**
20	1,123,370	1,125,845	1,124,927	1,123,329	**1,123,297**	**1,123,297**	**1,123,297**	**1,123,297**	**1,123,297**
30	1,683,154	1,688,954	1,684,232	**1,683,067**	–	1,683,139	1,683,154	**1,683,067**	1,683,154
40	2,242,678	2,249,518	2,245,261	2,242,596	**2,242,575**	2,242,882	2,242,882	2,242,596	2,242,882
50	2,800,717	2,805,663	2,803,892	**2,800,495**	–	2,800,889	2,801,445	**2,800,495**	2,800,743
60	3,360,492	3,365,694	3,361,457	3,360,027	**3,359,954**	3,360,316	3,360,699	3,360,181	3,361,119
70	3,921,101	3,924,225	3,924,405	**3,921,031**	–	3,921,331	3,922,412	**3,921,031**	3,921,111
80	4,480,798	4,483,632	4,483,871	4,480,379	–	4,480,515	4,480,617	**4,480,328**	4,480,617
90	5,039,429	5,045,894	5,045,587	**5,039,349**	–	5,041,434	5,040,025	5,039,421	5,040,020
100	**5,597,770**	5,605,045	5,602,364	5,597,843	**5,597,770**	5,597,962	5,600,133	5,597,964	5,598,876

The best found values for each test instance are boldfaced.

Table 4
Computational results: Case Study 2 [17].

Time horizon	24 h	72 h	168 h	24 h	72 h	168 h
Method	Operating cost (M$)			CPU time (s)		
DP [17]	210.5	–	–	24.00	–	–
LR [17]	209	–	–	7.00	–	–
SA [17]	207.8	–	–	1690.00	–	–
CLP [17]	208.1	–	–	10.00	–	–
FO [11]	207.8	–	–	–	–	–
MRCGA [52]	204.6	–	–	–	–	–
MACO [42]	200.46	–	–	111.90	–	–
FAPSO [46]	196.73	–	–	6.07	–	–
ASSA [43]	196.7	–	–	3.96	–	–
TFSA [45]	197.98	–	–	3.43	–	–
IPPDTM [2]	196.06	–	–	1.36	–	–
HASSA [44]	196.96	601.4	1410.47	5.01	9.04	37.64
EMO-ALHN [7]	197.5	590.66	1376.55	0.21	0.66	1.91
ALHN-LR [8]	195.87	585.27	1366.18	8.64	13.58	16.21
GVNS-R	194.44	583.62	1365.48	7.18	19.55	39.54
GVNS-G	194.05	583.71	1363.74	9.94	9.62	39.10
Adaptive_GVNS-R	194.16	583.76	1364.92	9.92	11.98	39.85
Adaptive_GVNS-G	**193.94**	**583.32**	**1362.41**	9.67	19.52	39.59

The best found values for each test instance are boldfaced.

Table 5
Computational Results with time limit set to 600s: Case Study 2 [17].

Time horizon	24 h	72 h	168 h	24 h	72 h	168 h
Method	Operating cost (M$)			CPU time (s)		
GVNS-R	**193.75**	581.58	1360.35	416.36	597.36	598.35
GVNS-G	**193.75**	582.26	1360.45	538.17	491.58	589.48
Adaptive_GVNS-R	**193.75**	**581.57**	1358.66	541.39	450.51	592.62
Adaptive_GVNS-G	**193.75**	**581.57**	**1358.65**	359.63	390.17	539.59

The best found values for each test instance are boldfaced.

From the results presented in Table 3 we conclude that all tested GVNS based heuristics are able to provide high quality solutions for all test instances. On test instances with 10 and 20 units, all GVNS variants as well as MILP based approach, succeeded in reaching optimal solutions. For all other instances, GVNS heuristics provide solutions very close to the corresponding best known values. Regarding the number of the best known solutions attained, we may conclude that Adaptive_GVNS-R, MILP based approach and CPLEX solver exhibit the best performances, while SBB solver and DICOPT solver exhibit the worst performances.

Comparison of GVNS approaches with other heuristics on instances from Case Study 2

In order to perform the comparison of our methods with other heuristic approaches on this data set, the start up cost in the first hour is neglected as in [2]. The results obtained by the heuristics from the literature are compared with our methods in Table 4: dynamic programming (DP) [17], Lagrangian relaxation (LR) [17], simulated annealing (SA) [17], constrained logic programming (CLP) [17], fuzzy optimization (FO) [11], matrix real coded genetic algorithm (MRCGA) [52], memory bounded ant colony optimization (MACO) [42], fuzzy adaptive particle swarm optimization (FAPSO) [46], absolutely stochastic simulated annealing (ASSA) [43], twofold simulated annealing (TFSA) [45], heuristic and ASA (HASSA) [44], enhanced merit order and augmented Lagrange Hopfield network (EMO-ALHN) [7], improved pre-prepared power demand and Muller method (IPPDTM) [2] and Augmented Lagrange hopfield network based Lagrangian relaxation (ALHN-LR) [8].

DP [17], LR [17], SA [17], and CLP [17] were executed on 486-66 PC, MRCGA [52] on Intel Celeron 1.2 GHz, ASSA [43] on Intel Pentium 4 1.4 GHz CPU, TFSA [45] and HASSA [44] on Intel(R) Celeron(TM) CPU, EMO-ALHN [7] on Intel Cleron 1.1 GHz, IPPDTM [2] on Pentium IV 2.8 GHz, ALHN-LR [8] on Intel Celeron 1.5 GHz. There is no report of computer used for the FO, MACO and FAPSO methods. GVNS approaches have been run on a computer with Intel i7 2.8 GHz CPU.

Among heuristics mentioned above, some have been tested on the same 38-units system, but with increased operating time. Namely, the total time of 24 h has been extended to 72 and 168 h. The increased load demands are adapted naturally. Such cases are compared in columns 3 and 4 of Table 4, i.e., only the costs of HASSA, EMO-ALHN, ALHN-LR and GVNS methods are given. In order to perform fair comparison with previous approaches, maximum CPU time allowed to be consumed by our GVNS methods were set to 10 s for time horizon of 24 h, 20 s for time horizon of 72 h and 40 s for time horizon of 168 h. However, in Table 5 we present results obtained by our GVNS methods extending the time limits to 600 s for each time horizon.

Computational results show that our methods, for any time horizon, provide better quality solutions than those obtained by previously proposed methods. It should be noted that for any time horizon solutions offered by Adaptive_GVNS-G are better than those found by other GVNS based methods. This result could be explained by the fact that the solution space is enormous, therefore it is important to start the exploration from a reasonably good initial solution (i.e., greedy solution) in order to get high quality solution within the imposed time limit. Additionally, the adaptive mechanism embedded within GVNS turns out to be powerful enough to help GVNS to provide high-quality solutions in a reasonable amount of time.

Comparison of GVNS approaches with exact methods on instance from Case Study 2

The exact methods were also applied for determining the optimal production of 38-units system over the time horizon of 24 h.

Table 6
Comparison with exact methods: Case Study 2 [17].

Method	B&C	SBB	DICOPT	CPLEX	GVNS-R	GVNS-G	Adaptive_GVNS-R	Adaptive_GVNS-G
No. of units	Operating cost ($)							
38	203,321,193	204,116,508	204,128,604	203,321,193	203,620,748	203,566,749	203,582,686	203,620,748

Table 7
Comparison of GVNS variants on instances with planing horizon of 72 h.

No. units	Best cost	GVNS-R			GVNS-G			Adaptive-GVNS-R			Adaptive-GVNS-G		
		Cost	dev. (%)	Time (s)	Cost	dev. (%)	Time (s)	Cost	dev. (%)	Time (s)	Cost	dev. (%)	Time (s)
10	**1,691,572**	**1,691,572**	0.000	5.87	**1,691,572**	0.000	5.81	**1,691,572**	0.000	5.81	**1,691,572**	0.000	5.73
20	**3,367,418**	**3,367,418**	0.000	26.81	**3,367,418**	0.000	126.65	**3,367,418**	0.000	176.77	**3,367,418**	0.000	24.77
30	**5,044,207**	5,044,451	0.005	116.58	5,044,249	0.001	61.40	**5,044,207**	0.000	87.34	5,044,249	0.001	43.17
40	**6,722,477**	6,725,219	0.041	233.81	6,724,626	0.032	126.95	6,722,741	0.004	276.84	**6,722,477**	0.000	436.47
50	**8,394,126**	8,394,738	0.007	513.82	8,394,821	0.008	541.01	**8,394,126**	0.000	367.40	8,394,675	0.007	298.51
60	**10,073,877**	10,074,667	0.008	420.50	10,075,536	0.016	373.20	**10,073,877**	0.000	487.80	10,074,634	0.008	432.42
70	**11,753,788**	11,757,361	0.030	525.55	11,755,044	0.011	417.61	**11,753,788**	0.000	294.25	11,755,772	0.017	593.52
80	**13,430,925**	**13,430,925**	0.000	579.04	13,432,944	0.015	571.96	13,431,550	0.005	501.32	13,432,518	0.012	578.78
90	**15,109,072**	15,112,686	0.024	380.58	15,112,848	0.025	541.26	15,111,091	0.013	572.21	**15,109,072**	0.000	598.70
100	**16,783,097**	16,784,132	0.006	572.83	16,784,628	0.009	583.78	**16,783,097**	0.000	454.15	16,784,365	0.008	560.90
Avg.	9237055.99	9238317.00	0.012	337.54	9238368.58	0.012	334.96	9237346.74	0.002	322.39	9237675.24	0.005	357.30

The best found values for each test instance are boldfaced.

Table 8
Comparison of GVNS variants on instances with planing horizon of 168 h.

No. units	Best cost	GVNS-R			GVNS-G			Adaptive-GVNS-R			Adaptive-GVNS-G		
		Cost	dev. (%)	Time (s)	Cost	dev. (%)	Time (s)	Cost	dev. (%)	Time (s)	Cost	dev. (%)	Time (s)
10	**3,946,841**	**3,946,841**	0.000	11.48	**3,946,841**	0.000	15.49	**3,946,841**	0.000	21.54	**3,946,841**	0.000	21.17
20	**7,855,658**	7,855,919	0.003	569.06	**7,855,658**	0.000	47.39	**7,855,658**	0.000	172.42	**7,855,658**	0.000	109.78
30	**11,766,439**	11,766,831	0.003	284.92	**11,766,439**	0.000	316.49	**11,766,439**	0.000	286.86	**11,766,439**	0.000	248.36
40	**15,684,448**	15,687,211	0.018	582.02	**15,684,448**	0.000	580.23	15,687,820	0.022	558.08	15,685,112	0.004	282.89
50	**19,583,473**	**19,583,473**	0.000	575.00	19,584,708	0.006	509.11	19,583,700	0.001	565.95	19,584,330	0.004	599.13
60	**23,504,315**	**23,504,315**	0.000	482.46	23,506,263	0.008	582.30	23,505,281	0.004	497.49	23,505,475	0.005	592.99
70	**27,426,852**	**27,426,852**	0.000	598.71	27,432,842	0.022	586.98	27,428,753	0.007	594.64	27,429,965	0.011	583.27
80	**31,340,087**	**31,340,087**	0.000	597.70	31,341,845	0.006	587.79	31,340,474	0.001	591.40	31,341,820	0.006	593.78
90	**35,259,360**	**35,259,360**	0.000	590.48	35,261,510	0.006	587.69	35,262,007	0.008	591.80	35,260,891	0.004	537.39
100	**39,162,034**	39,163,354	0.003	598.53	39,170,260	0.021	592.91	**39,162,034**	0.000	593.68	39,167,165	0.013	570.10
Avg.	21552950.71	21553424.29	0.003	489.04	21555081.39	0.007	440.64	21553900.65	0.004	447.38	21554369.67	0.005	413.89

The best found values for each test instance are boldfaced.

According to reported objective function values, it can be concluded that they did not neglect start up cost in the first hour since these values are much greater than those of recently proposed methods. For that reason, we have also included the start up cost in the first hour in the value of objective function. Results obtained by GVNS methods, B&C, SBB solver, DICOPT solver, CPLEX solver are given in Table 6. The best known objective function values for this test instance is provided by B&C and CPLEX solver. The values of objective function found by GVNS methods are about 0.15% greater than the best known value. On the other hand, those values are significantly less than that provided by SBB solver or DICOPT solver.

Computational results for time horizons longer than 24 h

Most solvers can handle up to around 100 generators within 24 time periods (h). However, the demand for units schedules over a longer time horizon is required in reality. (Or, if the time step is less then one hour, then the same issue is evoked - an instance too large for solver to deal with.) For that purpose, we have generated test instances with time horizons of 72 and 168 time periods. Each of those instances is derived by accordingly extending load demand of each test instances from the Case Study 1. The computational results obtained by proposed GVNS variants are given in Tables 7 and 8. On each test instance, for each GVNS variant we report the following values:

- Solution value in *Cost* column.
- CPU time consumed to find the solution in *Time* column.
- Percentage deviation of the reported solution value from the best found value regarding all GVNS variants (given in column *Best Cost*).

From the results presented in Tables 7 and 8 we may conclude that objective function values found by all four GVNS variants are very similar for all test instances (the maximal deviation of a solution value, reported by a GVNS variant, from the best known solution value is not greater than 0.05%). More detailed observations are as follows:

- All GVNS variants except GVNS-R provide the same solutions on instances with 10 and 20 units. However, on the instance with 10 units and time horizon of 72 h, GVNS-R succeeds to find same solution as the other GVNS variants.
- Almost all best reported solutions are found by either Adaptive_GVNS-R or GVNS-R. Namely, these two approaches together offer best solutions for 17 out of 20 instances.

Table 9
Fuel cost data of 10 units system.

U	a_i	b_i	c_i	$P_{i,min}$	$P_{i,max}$	Cold start cost ($)	Hot Start cost ($)	$T_{i,up}$	$T_{i,down}$	Cold start (h)	Initial status (h)
1	1000	16,19	0.00048	150	455	4500	9000	8	8	5	8
2	970	17.26	0.00031	150	455	5000	10,000	8	8	5	8
3	700	16.6	0.002	20	130	550	1100	5	5	4	−5
4	680	16.5	0.00211	20	130	560	1120	5	5	4	−5
5	450	19.7	0.00398	25	162	900	1800	6	6	4	−6
6	370	22.26	0.00712	20	80	170	340	3	3	2	−3
7	480	27.74	0.00079	25	85	260	520	3	3	2	−3
8	660	25.92	0.00413	10	55	30	60	1	1	0	−1
9	665	27.27	0.00222	10	55	30	60	1	1	0	−1
10	670	27.79	0.00173	10	55	30	60	1	1	0	−1

Table 10
Load demand for 10 units system.

Hour	1	2	3	4	5	6	7	8	9	10	11	12
Demand (MW)	700	750	850	950	1000	1100	1150	1200	1300	1400	1450	1500
Hour	13	14	15	16	17	18	19	20	21	22	23	24
Demand (MW)	1400	1300	1200	1050	1000	1100	1200	1400	1300	1100	900	800

Table 11
Fuel cost data of 38 units system.

U	a_i	b_i	c_i	$P_{i,min}$	$P_{i,max}$	Start up cost ($)	$T_{i,up}$	$T_{i,down}$
1	64,782	796.9	0.3133	220	550	805,000	18	8
2	64,782	796.9	0.3133	220	550	805,000	18	8
3	64,670	795.5	0.3127	200	500	805,000	18	8
4	64,670	795.5	0.3127	200	500	805,000	18	8
5	64,670	795.5	0.3127	200	500	805,000	18	8
6	64,670	795.5	0.3127	200	500	805,000	18	8
7	64,670	795.5	0.3127	200	500	805,000	18	8
8	64,670	795.5	0.3127	200	500	805,000	18	8
9	172,832	915.7	0.7075	200	500	402,500	7	7
10	172,832	915.7	0.7075	114	500	402,500	7	7
11	176,003	884.2	0.7515	114	500	402,500	7	7
12	173,028	884.2	0.7083	114	500	402,500	7	7
13	91,340	1250.1	0.4211	110	500	575,000	9	8
14	63,440	1298.6	0.5145	90	365	575,000	12	8
15	65,468	1298.6	0.5691	82	365	575,000	12	8
16	72,282	1290.8	0.5691	120	325	575,000	10	8
17	190,928	238.1	25.881	65	315	23,000	1	1
18	285,372	1149.5	38.734	65	315	23,000	1	1
19	271,376	1269.1	36.842	65	315	23,000	1	1
20	39,197	696.1	0.4921	120	272	575,000	9	8
21	45,576	660.2	0.5728	120	272	575,000	9	8
22	28,770	803.2	0.3572	110	260	460,000	11	8
23	36,902	818.2	0.9415	80	190	92,000	14	7
24	105,510	33.5	52.123	10	150	23,000	1	1
25	22,233	805.4	11.421	60	125	115,000	8	8
26	30,953	707.1	20.275	55	110	287,500	14	7
27	17,044	833.6	30.744	35	75	253,000	14	7
28	81,079	2188.7	16.765	20	70	5750	1	1
29	124,767	1024.4	26.355	20	70	5750	1	1
30	121,915	837.1	30.575	20	70	5750	1	1
31	120,780	1305.2	25.098	20	70	5750	1	1
32	104,441	716.6	33.722	20	60	7670	1	1
33	83,224	1633.9	23.915	25	60	7670	1	1
34	111,281	969.5	32.562	18	60	7670	1	1
35	64,142	2625.8	18.362	8	60	7670	1	1
36	103,519	1633.9	23.915	25	60	7670	1	1
37	13,547	694.7	8.482	20	38	69,000	11	8
38	13,518	655.9	9.693	20	38	69,000	11	8

Table 12
Load demand for 38 units system.

Hour	1	2	3	4	5	6	7	8	9	10	11	12
Demand (MW)	5700	5400	5150	4850	4950	4800	4850	5400	6700	7850	8000	8100
Hour	13	14	15	16	17	18	19	20	21	22	23	24
Demand (MW)	6900	8150	8250	8000	7800	7100	6800	7300	7100	6800	6550	6450

- On instances with planing horizon of 72 h `Adaptive_GVNS-R` outperforms all the others regarding both solution quality and average CPU time needed to provide a solution for an instance.
- Regarding the average solution cost achieved by each of compared GVNS heuristics on instances with planing horizon of 168 h, we conclude that `GVNS-R` outperforms all the others. The second best heuristic turns to be `Adaptive_GVNS-R`, while `Adaptive_GVNS-G` takes the third place in the overall ranking. Finally, `GVNS-G` is ranked as the worst heuristic.
- On average, all proposed GVNS variants consume similar amount of CPU time to solve an instance with planing horizon of 72 h. On the other hand, on the instances with planing horizon of 168 h, it appears that `GVNS-R` is the slowest, whereas `Adaptive_GVNS-G` is the fastest GVNS variant. The remaining two GVNS variants spend almost the same amount of CPU time, on average, to solve an instance.

Concluding remarks

The main contribution of this paper is suggestion of a novel methods, based on General Variable Neighborhood Search (GVNS) for solving the unit commitment problem (UCP). So far numerous metaheuristics have been proposed for solving the UCP, but not GVNS. We propose an adaptive mechanism within GVNS which helps to decide what neighborhood structure to apply in some stage of the solution process. The computational results show that the proposed GVNS methods, with and without the adaptive mechanism, prove to be very efficient in solving the UCP, regarding both CPU times spent and result quality. We compare our new UCP heuristics with more than 20 successful methods from the literature. Furthermore, proposed heuristics are able to solve large size instances with time horizons up to one week, which is the largest time horizon considered in the literature.

Proposed methods have been tested in solving general convex UCP. However, they can be easily adapted for solving UCP with non-convex objectives. It would be sufficient to use another method for solving economic dispatch problem instead of one that we use here, i.e., the lambda iteration method.

Thus, future research may include developing new GVNS and Variable Neighborhood Decomposition Search (VNDS) based methods for solving the UCP and related UCPs with additional constraints such as ramp rate constraints, environmental constraints, and emission constraints.

Acknowledgement

This work was conducted at the National Research University Higher School of Economics, Nizhni Novgorod, Russia and supported by RSF grant 14-41-00039.

Appendix A

See Tables 9–12.

References

[1] Burns RM, Gibson CA. Optimization of priority list for a unit commitment problem. IEEE PES Summer Meeting 1975;75:453–61.
[2] Chandram K, Subrahmanyam N, Sydulu M. Unit Commitment by improved pre-prepared power demand table and Muller method. Int J Electr Power Energy Syst 2011;33:106–14.
[3] Cheng CP, Liu CW, Liu CC. Unit commitment by Lagrangian relaxation and genetic algorithm. IEEE Trans Power Syst 2000;15:707–14.
[4] Chung C, Yu H, Wong KP. An advanced quantum-inspired evolutionary algorithm for unit commitment. IEEE Trans Power Syst 2011;26:847–54.
[5] Damousis IGS, Bakirtziz AG, Dokopoulos PS. A solution to the unit commitment problem using integer coded genetic algorithm. IEEE Trans Power Syst 2004;19:1165–72.
[6] Dieu V, Ongsakul W. Enhanced augmented Lagrangian hopfield network for unit commitment. IEE Proc Gener Transm Distrib 2006;153:624–32.
[7] Dieu V, Ongsakul W. Enhanced merit order and augmented Lagrangian hopfield network for ramp rate constrained unit commitment. In: Proc of IEEE power system society meeting. Canada; 2006.
[8] Dieu V, Ongsakul W. Augmented Lagrange hopfield network based Lagrangian relaxation for unit commitment. Int J Electr Power Energy Syst 2011;33:522–30.
[9] Dillon JD, Walsh MP, OMalley MJ. Initialization of the augmented Hopfield network for improved generator scheduling. IEE Proc Gener Transm Distrib 2002;149:593–9.
[10] Ebrahimi J, Hosseinian S, Gharehpetian G. Unit commitment problem solution using shuffled frog leaping algorithm. IEEE Trans Power Syst 2011;26:573–81.
[11] El-Saadawi MM, Tantawi MA, Tawfik E. A fuzzy optimization-based approach to large scale thermal unit commitment. Electric Power Syst Res 2004;72:245–52.
[12] Feo TA, Resende MG. Greedy randomized adaptive search procedures. J Global Optim 1995;6:109–33.
[13] Hansen P, Mladenović N. Simultaneous static unit commitment and economic dispatch by dynamic programming. Cahiers du GERAD, G-96-26; 1996.
[14] Hansen P, Mladenović N. A separable approximation dynamic programming algorithm for economic dispatch with transmission losses. Yugoslav J Oper Res 2002;12:157–66.
[15] Hansen P, Mladenović N. Developments of variable neighborhood search. US: Springer; 2002. p. 415–39.
[16] Hansen P, Mladenović N, Moreno-Pérez JA. Variable neighbourhood search: methods and applications (invited survey). Ann Oper Res 2010;175:367–407.
[17] Huang KY, Yang HT, Huang CL. A new thermal unit commitment approach using constraint logic programming. IEEE Trans Power Syst 1998;13:936–45.
[18] Hu B, Raidl G. Variable neighborhood descent with self-adaptive neighborhood-ordering. In: Cotta C, Fernandez AJ, Gallardo JE. editors. Proceedings of the 7th EU/MEeting on adaptive, self-adaptive, and multi-level metaheuristics. Malaga, Spain; 2006.
[19] Jabr RA. Rank-constrained semidefinite program for unit commitment. Int J Electr Power Energy Syst 2013;47:13–20.
[20] Jeong YW, Park JB, Jang SH, Lee K. A new quantum inspired binary PSO: application to unit commitment problems for power systems. IEEE Trans Power Syst 2010;25:1486–95.
[21] Juste KA, Kita H, Tanaka E, Hasegawa J. An evolutionary programming solution to the unit commitment problem. IEEE Trans Power Syst 1999;14:1452–9.
[22] Kazarlis A, Bakirtzis AG, Petridis V. A genetic algorithm solution to the unit commitment problem. IEEE Trans Power Syst 1996;11:83–92.
[23] Kritzinger S, Doerner KF, Tricoire F, Hartl RF. Adaptive search techniques for problems in vehicle routing, part I: a survey. Yugoslav J Oper Res 2014. http://dx.doi.org/10.2298/YJOR140217009K.
[24] Kritzinger S, Doerner KF, Tricoire F, Hartl RF. Adaptive search techniques for problems in vehicle routing, part II: a numerical comparison. Yugoslav J Oper Res 2014. http://dx.doi.org/10.2298/YJOR140217011K.
[25] Lau T, Chung C, Wong K, Chung T, Ho S. Quantum-inspired evolutionary algorithm approach for unit commitment. IEEE Trans Power Syst 2009;24:1503–12.
[26] Lazić J, Todosijević R, Hanafi S, Mladenović N. Variable and single neighbourhood diving for MIP feasibility. Yugoslav J Oper Res 2014. http://dx.doi.org/10.2298/YJOR140417027L.
[27] Lowery PG. Generating unit commitment by dynamic programming. IEEE Trans Power Syst 1966;85:4224266.
[28] Marcovecchio MG, Novais AQ, Grossmann IE. A branch and bound search for the deterministic optimization of the thermal unit commitment problem. Part II: computational results; 2011. <http://egon.cheme.cmu.edu/Papers/Marcovecchio-Novais-Grossmann-PartII.pdf>.
[29] Mhanna SN, Jabr RA. Application of semidefinite programming relaxation and selective pruning to the unit commitment problem. Electric Power Syst Res 2012;90:85–92.
[30] Mjirda A, Todosijević R, Hanafi S, Hansen P, Mladenović N. Sequential variable neighborhood descent variants: an empirical study on Travelling salesman problem. Cahiers du GERAD, G-2015-37; 2015.

[31] Mladenović N, Hansen P. Variable neighborhood search. Comput Oper Res 1997;24:1097–100.
[32] Mladenović N, Todosijević R, Urošević D. An efficient general variable neighborhood search for large TSP problem with time windows. Yugoslav J Oper Res 2012;22:141–51.
[33] Moghimi Hadji M, Vahidi B. A solution to the unit commitment problem using imperialistic competition algorithm. IEEE Trans Power Syst 2012;27:117–24.
[34] Ongsakul W, Petcharaks N. Unit commitment by enhanced adaptive Lagrangian relaxation. IEEE Trans Power Syst 2004;19:620–8.
[35] Pang CK, Sheble GB, Albu F. Evaluation of dynamic programming based methods and multiple area representation for thermal unit commitment. IEEE Trans Power Appl Syst 1981;100:1212–8.
[36] Polacek M, Benkner S, Doerner KF, Hartl RF. A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. BuR-Bus Res 2008;1(2):207–18.
[37] Quan R, Jian J, Mu Y. Tighter relaxation method for unit commitment based on second-order cone programming and valid inequalities. Int J Electr Power Energy Syst 2014;55:82–90.
[38] Rajan CCA, Mohan MR. An evolutionary programming based Tabu search method for solving the unit commitment problem. IEEE Trans Power Syst 2004;19:577–85.
[39] Rebennack S, Pardalos PM, Pereira MVF, Iliadis NA. Handbook of power systems. New York: Springer; 2010.
[40] Ripon KSN, Glette K, Khan KN, Hovin M, Torresen J. Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities. Swarm Evol Comput 2013;8:1–12.
[41] Roy K. Solution of unit commitment problem using gravitational search algorithm. Int J Electr Power Energy Syst 2013;53:85–94.
[42] Saber AY, Alshareef AM. Scalable unit commitment by memory-bounded ant colony optimization with A∗ local search. Int J Electr Power Energy Syst 2008;30:403–14.
[43] Saber AY, Senjyu T, Miyagi T, Urasaki N, Funabashi T. Fuzzy unit commitment scheduling using absolutely stochastic simulated annealing. IEEE Trans Power Syst 2006;21:955–64.
[44] Saber AY, Senjyu T, Miyagi T, Urasakiand N, Funabashi T. Unit commitment by heuristics and absolutely stochastic simulated annealing. IET Gener Transm Distrib 2007;1:234–43.
[45] Saber AY, Senjyu T, Yona A, Urasaki N, Funabashi T. Fuzzy unit commitment solution – a novel twofold simulated annealing approach. Electr Power Syst Res 2007;77:1699–712.
[46] Saber AY, Senjyu T, Yona A, Funabashi T. Unit commitment computation by fuzzy adaptive particle swarm optimisation. IET Gener Transm Distrib 2007;1:456–65.
[47] Sasaki H, Watanabe M, Yokoyama R. A solution method of unit commitment by artificial neural networks. IEEE Trans Power Syst 1992;7:974–81.
[48] Senjyu T, Yamashiro H, Shimabukuro K, Uezato K, Funabashi T. A unit commitment problem by using genetic algorithm based on characteristic classification. IEEE/Power Eng Soc Winter Meet 2002;1:58–63.
[49] Silva Jr I, Carneiro Jr S, De Oliveira EJ, Pereira JLR, Garcia PAN, Marcato ALM. A Lagrangian multiplier based sensitive index to determine the unit commitment of thermal units. Int J Electr Power Energy Syst 2008;30:504–10.
[50] Srinivasan D, Chazelas J. A priority list based evolutionary algorithm to solve large scale unit commitment problem. In: International conference on power system technology Powercon 2004. Singapore; 2004. p. 21–4.
[51] Stenger A, Vigo D, Enz S, Schwind M. An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. Transp Sci 2013;47:64–80.
[52] Sun L, Zhang Y, Jiang C. A matrix real-coded genetic algorithm to the unit commitment problem. Electr Power Syst Res 2006;76:716–28.
[53] Swarup KS, Yamashiro S. Unit commitment solution methodology using genetic algorithm. IEEE Trans Power Syst 2002;17:87–91.
[54] Todosijević R, Mladenović M, Hanafi S, Crévits I. VNS based heuristic for solving the unit commitment problem. Electr Notes Discrete Math 2012;39:153–60.
[55] Viana A, Pedroso JP. A new MILP-based approach for Unit Commitment in power production planning. Int J Electr Power Energy Syst 2013;44:997–1005.
[56] Zhao B, Guo CX, Bai BR, Cao YJ. An improved particle swarm optimization algorithm for unit commitment. Int J Electr Power Energy Syst 2006;44:432–512.
[57] Zheng QP, Wang J, Pardalos PM, Guan Y. A decomposition approach to the two-stage stochastic unit commitment problem. Ann Oper Res 2013;210 (1):387–410.
[58] Wood AJ, Wollenberg BF. Power generation, operation and control. 2nd revised ed. New York: Wiley; 1996.