Les Cahiers du GERAD ISSN: 0711–2440

Adaptive general variable neighborhood search heuristics for solving unit commitment problem

R. Todosijević, M. Mladenović,
S. Hanafi, N. Mladenović, I. Crévits

G–2015–43

April 2015

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Adaptive general variable neighborhood search heuristics for solving unit commitment problem

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

[a] *LAMIH – Université de Valenciennes et du Hainaut-Cambrésis, 59313 Valenciennes Cedex 9, France*

[b] *GERAD, HEC Montréal, Montréal (Québec) Canada, H3T 2A7*

racatodosijevic@gmail.com
mladja87@gmail.com
said.hanafi@univ-valenciennes.fr
nenad.mladenovic@gerad.ca
igor.crevits@univ-valenciennes.fr

April 2015

Les Cahiers du GERAD
G–2015–43

Abstract: Unit commitment problem (UCP) for thermal units consists of finding an optimal electricity production plan for a long time horizon. In this paper we propose a hybrid approaches which combine variable neighborhood search 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 state-of-the-art approaches.

Key Words: Power systems, unit commitment problem, mixed integer nonlinear problem, variable neighborhood search.

Acknowledgments: This work is conducted at National Research University Higher School of Economics and supported by RSF grant 14-41-00039.

1 Introduction

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.

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 can not 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 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, 47, 52], tabu search algorithms [38], particle swarm optimization algorithms [45, 56], ant colony algorithms [41], fuzzy logic [11], artificial neural networks [9, 46], evolutionary programming [21], simulated annealing [42–44].

In this paper we propose a hybrid approaches that combine Variable Neighborhood Search (VNS) metaheuristic with mathematical programming. We in fact substantially extend our conference paper [53] by considering 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 that cannot be solved by exact solvers. Computational results show that the proposed heuristic outperforms all current heuristic approaches, while improving running times for most instances. It is especially true for largest size instances.

The rest of the paper is organized as follows. In Section 2 we provide mathematical formulation of UCP, in Section 3 we describe our method. Section 4 presents comparison of our method with existing approaches, while Section 5 concludes the paper and offers directions for future research.

2 Problem formulation

2.1 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

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

$$P_i^{min} \leq P_i \leq P_i^{max}, \qquad \text{for } \; i = 1, \ldots, n \tag{2}$$

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 [57]

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

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

2.1.1 Lambda iteration method for solving economic dispatch problem

The lambda-iteration method [57] 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 stopes 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`();
1 $\lambda^{min} \leftarrow \min_{i=1,\ldots,n} \frac{dF_i(P_i^{min})}{dP_i}$;
2 $\lambda^{max} \leftarrow \max_{i=1,\ldots,n} \frac{dF_i(P_i^{max})}{dP_i}$;
3 $\epsilon \leftarrow 10^{-6}$;
repeat
4 $\quad \lambda \leftarrow (\lambda^{min} + \lambda^{max})/2$;
5 $\quad$ Calculate P_i from $\frac{dF_i(P_i)}{dP_i} = \lambda$;
6 $\quad \Delta = P^D - \sum_{i=1}^{n} P_i$;
7 $\quad$ **if** $P^D > \sum_{i=1}^{n} P_i$ **then** $\lambda^{min} = \lambda$;
8 $\quad$ **if** $P^D < \sum_{i=1}^{n} P_i$ **then** $\lambda^{max} = \lambda$;
until $|\Delta| \leq \epsilon$;
return $P_1, \ldots, P_n$;

2.2 Unit commitment problem

The basic goal of 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:

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

subject to:

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

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

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

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

where

$$ST_i = \begin{cases} HSC_i, & \text{if } T_{i,down} \leq T_{i,off}^t \leq T_{i,cold} + T_{i,down} \\ CSC_i, & \text{otherwise} \end{cases} \tag{9}$$

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 Eq. (8) represents the minimum up and down times 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 ($T_{i,up}$, $T_{i,down}$). These two constraints represent the time constraints, while (5), (6) and (7) are production constraints. This means that UC 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).

3 General variable neighborhood search for solving UCP

As we mentioned earlier, finding an optimal solution for large size 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 heuristic [16,31].

VNS is a flexible framework for building heuristics for approximately solving 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 optimal for another neighborhood structure; (ii) A global optimum is a local optimum with respect to all neighborhood structures; (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.

3.1 Neighborhood structures

Suppose that a feasible solution is represented by matrix U. Let us define with $N_k^{'}(U)$ the set of all solutions which can be obtained by changing values exactly k elements of the matrix U. Obviously, such 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 $N_k^{'}(U)$.

3.2 Generating 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

$$M_j = \frac{2 \cdot F((P_{j,max} + P_{j,min})/2)}{P_{j,max} + P_{j,min}} \tag{10}$$

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 unfeasible, 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:

$$T_{i,on}^{t} = \begin{cases} T_{i,on}^{t-1} + 1 & \text{if } U_{i,t} = 1 \\ 0 & \text{otherwise} \end{cases}$$
$$T_{i,off}^{t} = \begin{cases} T_{i,off}^{t-1} + 1 & \text{if } U_{i,t} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

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 decommit redundant units due to the minimum up and down time repairing, thereby reducing the operating cost. The algorithm searches for units that can be decommitted without violating constraints starting from the unit with highest values of M_i until there is no unit that can be decommitted.

Algorithm 2: Primary unit scheduling

```
Function Primary();
Calculate values M_i according to (10)
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} ← 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 (11);
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_{i,on}^{t-1} < T_{i,up}) then U_{i,t} = 1;
        if (U_{i,t} = 0 and U_{i,t-1} = 1 and T_{i,off}^{t+T_{i,down}-1} < T_{i,down}) then U_{i,t} = 1;
        Update the on/off status for the unit i in (11);
    end
end
return U;
```

Algorithm 4: Deccomitment of excessive units

```
Function Deccomit(U);
for t=1 to T do
    E = ∅ ;
    for i=1 to N do
        if (unit i can be decommitted without violations) then E = E ∪ {i};
    end
    repeat
        Choose k ∈ E with the highest value M_k ;
        if (unit k can be decommitted without violating spinning reserves constraint) then
        U_{k,t} ← 0;
        E = E \ {k};
    until E ≠ ∅;
end
return U;
```

Algorithm 5: Building greedy solution

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

3.3 Pipe variable neighborhood descents

Variable neighborhood descent (VND) is deterministic variant of VNS where neighborhoods are ordered in a sequence and used one after another until a local minimum with respect to all of them is reached. Usually 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) [54]. In this paper we develop two variants of pipeVND for solving UC that differ in order of the local searches during the optimization process. However, both pipeVNDs consist of the same set of two neighborhoods sequentially explored one after another. Additionally, each of them are iterated until there is no improvement in objective function value (Algorithm 8 and Algorithm 9). The used local searches attempt to decommit units, preserving feasibility. They are based on the priority list, i.e., the search for units which will be decommited 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 decommit a unit. First local search (Algorithm 6) attempts to decommit unit i for a period of T_i^{down} hours, the second (Algorithm 7), attempts to decommit each unit in one hour. 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.

3.4 General variable neighborhood search algorithms

Until now, many variants of VNS have been proposed in the literature (see e.g. [15,16,26] for recent surveys). However, the widely used variant is 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. [18,23,24,36,39,50]).

For solving Unit Commitment problem we developed two variants of GVNS that uses the same shaking procedure `Shake`(U,k) for diversification. The function `Shake`(U,k) at output returns a random solution from the k-th neighborhood of a given solution U. 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

Algorithm 6: Local search 1

Function `LS1`(U);
1 $E = \{i_1, i_2, \dots i_N\}$ /* indices of units sorted according to descending merit order */;
2 $U^{'} \leftarrow U$;
3 **for** *k=1* **to** N **do**
4 **for** *t=1* **to** T-T_i^{down}+*1* **do**
if *Unit* i_k *can be decommitted during time period* [*t*,*t*+T_i^{down}-*1*] *keeping feasibility* **then**
5 **for** $h = t$ **to** $t + T_i^{down}$-1 **do** $U^{'}_{i_k,h} \leftarrow 0$;
6 **if** $\mathrm{G}(U^{'}) < G(U)$ **then** $U \leftarrow U^{'}$;
7 **else** $U^{'} \leftarrow U$;
end
end
end
return U;

Algorithm 7: Local search 2

Function `LS1`(U);
1 $E = \{i_1, i_2, \dots i_N\}$ /* indices of units sorted according to descending merit order */;
2 $U^{'} \leftarrow U$;
3 **for** *k=1* **to** N **do**
4 **for** *t=1* **to** T **do**
if *Unit* i_k *can be decommitted in hour t keeping feasibility* **then**
5 $U^{'}_{i_k,t} \leftarrow 0$;
6 **if** $\mathrm{G}(U^{'}) < G(U)$ **then** $U \leftarrow U^{'}$;
7 **else** $U^{'} \leftarrow U$;
end
end
end
return U;

Algorithm 8: pipeVND1

Function `pipeVND1`(U);
repeat
1 $U \leftarrow$ `LS1(U)` ;
2 $U \leftarrow$ `LS2(U)` ;
until *There is no improvement*;
return U;

Algorithm 9: pipeVND2

Function `pipeVND2`(U);
repeat
1 $U \leftarrow$ `LS2(U)` ;
2 $U \leftarrow$ `LS1(U)` ;
until *There is no improvement*;
return U;

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 Algorithm 10 and Algorithm 11, respectively.

The value of VNS parameter k_{max} is set to 5, whereas the time limit is set to t_{max}=600 seconds for all developed heuristics.

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 f(U'') < f(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 f(U'') < f(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;
```

4 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 a new current solution. The whole process is repeated $N \cdot T$ times.

4.1 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 for 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.

4.2 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: Lagragian relaxation(LR) [22]; genetic algorithm (GA) [22]; enhanced adaptive Lagrangian relaxation (ELR) [22]; Dynamic Programming with ELR (DPLR) [22]; Lagrangian relaxation and genetic algorithm (LRGA) [3]; genetic algorithm based on characteristic classification (GACC) [47]; evolutionary programming (EP) [21]; priority-list-based evolutionary algorithm (PLEA) [49]; extended priority list (EPL) [49]; integer coded genetic algorithm (ICGA) [5]; a Lagrangian multiplier based sensitive index to determine the unit commitment of thermal units (LMBSI) [48]; 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) [40]; semidefinite programming (SDP) [19, 30]; tighter relaxation method (RM) [37]. Comparative results are given in Table 1. It should be emphasized that for test instance with 20 units, LRGA, SFLA and GSA heuristics report solution values better than the optimal solution value (which equals to 1123297 see [55]). Therefore, we boldfaced that value in Table 1, but values better than optimal present in italic font. For instance with 40 units the optimal solution is not known. Hence, it is questionable if the value of 2242178 obtained by LRGA is really reliable (since it reported better value than optimal for N=20).

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 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 [30].

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	
Method	Operating Cost($)						Average
LR [22]	565825	1130660	2258503	3394066	4526022	5657277	2922058.83
ELR [34]	563977	**1123297**	2244237	3363491	4485633	5605678	2897718.83
LRGA [3]	564800	*1122622*	**2242178**	3371079	4501844	5613127	2902608.33
DPLR [34]	564049	1128098	2256195	3384293	4512391	5640488	2914252.33
GA [22]	565825	1126243	2251911	3376625	4504933	5627437	2908829.00
GACC [47]	563977	1125516	2249715	3375065	4505614	5626514	2907733.50
EP [21]	564551	1125494	2249093	3371611	4498479	5623885	2905518.83
ICGA [5]	566404	1127244	2254123	3378108	4498943	5630838	2909276.67
PLEA [49]	563977	1124295	2243913	3363892	4487354	5607904	2898555.83
EPL [49]	563977	1124369	2246508	3366210	4489322	5608440	2899804.33
LMBSI [48]	563977	1123990	2243708	3362918	4483593	5602844	2896838.33
IPPDTM [2]	563977	-	2247162	3366874	4490208	5609782	-
QBPSO [20]	563977	**1123297**	2242957	3361980	4482085	5602486	2896130.33
QEA-UC [25]	**563938**	1123607	2245557	3366676	4488470	5609550	2899633.00
IQEA-UC [4]	**563938**	**1123297**	2242980	3362010	4482826	5602387	2896239.67
SFLA [10]	564769	*1123261*	2246005	3368257	4503928	5624526	2905124.33
ICA [33]	**563938**	1124274	2247078	3371722	4497919	5617913	2903807.33
GSA [40]	**563938**	*1123216*	2242741	3362447	4483864	5600883	2896181.50
SDP [19]	563938	1124357	2243328	3363031	4484365	5602538	2896926.17
SDP [30]	563977	1124410	2243144	3360512	4480652	5598727	2895237.00
RM [37]	563977	1123990	2243676	3361589	4481833	5599761	2895804.33
`GVNS-R`	563938	**1123297**	2242882	3360316	4480515	**5597962**	2894818.33
`GVNS-G`	563938	**1123297**	2242882	3360699	4480617	5600133	2895261.00
`Adaptive_GVNS-R`	563938	**1123297**	**2242596**	**3360181**	**4480328**	5597964	2894717.33
`Adaptive_GVNS-G`	563938	**1123297**	2242882	3361119	4480617	5598876	2895121.50

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 [48], IPPDTM [2], QBPSO [20], QEA-UC [25], IQEA-UC [4], GSA [40], 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. Notice that our code is run on a single processor while some other use more parallel CPUs.

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	
Method	CPU time(s)						Average
LMBSI [48]	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 [40]	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

4.3 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 [29] and MILP-based approach [55]. The comparison is presented in the 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	**563938**	572468	**563938**	564189	**563938**	**563938**	**563938**	**563938**	**563938**
20	1123370	1125845	1124927	1123329	**1123297**	**1123297**	**1123297**	**1123297**	**1123297**
30	1683154	1688954	1684232	**1683067**	-	1683139	1683154	**1683067**	1683154
40	2242678	2249518	2245261	2242596	**2242575**	2242882	2242882	2242596	2242882
50	2800717	2805663	2803892	**2800495**	-	2800889	2801445	**2800495**	2800743
60	3360492	3365694	3361457	3360027	**3359954**	3360316	3360699	3360181	3361119
70	3921101	3924225	3924405	**3921031**	-	3921331	3922412	**3921031**	3921111
80	4480798	4483632	4483871	4480379	-	4480515	4480617	**4480328**	4480617
90	5039429	5045894	5045587	**5039349**	-	5041434	5040025	5039421	5040020
100	**5597770**	5605045	5602364	5597843	**5597770**	5597962	5600133	5597964	5598876

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 best known solutions achieved we may conclude that `Adaptive_GVNS-R`, MILP based approach and CPLEX are the best methods. On the other hand, the worst two methods are SBB solver and DICOPT who provide the smallest number of the best known solutions.

4.4 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) [51], memory bounded ant colony optimization (MACO) [41], fuzzy adaptive particle swarm optimization(FAPSO) [45], absolutely stochastic simulated annealing (ASSA) [42], twofold simulated annealing (TFSA) [44], heuristic and ASA (HASSA) [43], 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 [51] on Intel Celeron 1.2 GHz, ASSA [42] on Intel Pentium 4 1.4 GHz CPU, TFSA [44] and HASSA [43] 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 hours has been extended to 72 and 168 hours. 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 seconds for time horizon of 24 hours, 20 seconds for time horizon of 72 hours and 40 seconds for time horizon of 168 hours. However, in Table 5 we present results obtained by our GVNS methods extending the time limits to 600 seconds 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.

Table 4: Computational results: Case study 2 [17]

Time horizon	24h	72h	168h	24h	72h	168h
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 [51]	204.6	-	-	-	-	-
MACO [41]	200.46	-	-	111.90	-	-
FAPSO [45]	196.73	-	-	6.07	-	-
ASSA [42]	196.7	-	-	3.96	-	-
TFSA [44]	197.98	-	-	3.43	-	-
IPPDTM [2]	196.06	-	-	1.36	-	-
HASSA [43]	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

Table 5: Computational results with time limit set to 600s: Case study 2 [17]

Time horizon	24h	72h	168h	24h	72h	168h
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

4.5 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 24h. 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.

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	203321193	204116508	204128604	203321193	203620748	203566749	203582686	203620748

4.6 Computational results for time horizons longer than 24h

Most solvers can handle up to around 100 generators within 24 time periods (hours). However, the demand for units schedules over a longer time horizon is required in reality. 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 Table 7 and Table 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*).

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

No.		GVNS-R			GVNS-G			Adaptive-GVNS-R			Adaptive-GVNS-G		
Units	Best cost	Cost	dev. (%)	Time(s)	Cost	dev. (%)	Time(s)	Cost	dev. (%)	Time(s)	Cost	dev. (%)	Time(s)
10	**1691572**	**1691572**	0.000	5.87	**1691572**	0.000	5.81	**1691572**	0.000	5.81	**1691572**	0.000	5.73
20	**3367418**	**3367418**	0.000	26.81	**3367418**	0.000	126.65	**3367418**	0.000	176.77	**3367418**	0.000	24.77
30	**5044207**	5044451	0.005	116.58	5044249	0.001	61.40	**5044207**	0.000	87.34	5044249	0.001	43.17
40	**6722477**	6725219	0.041	233.81	6724626	0.032	126.95	6722741	0.004	276.84	**6722477**	0.000	436.47
50	**8394126**	8394738	0.007	513.82	8394821	0.008	541.01	**8394126**	0.000	367.40	8394675	0.007	298.51
60	**10073877**	10074667	0.008	420.50	10075536	0.016	373.20	**10073877**	0.000	487.80	10074634	0.008	432.42
70	**11753788**	11757361	0.030	525.55	11755044	0.011	417.61	**11753788**	0.000	294.25	11755772	0.017	593.52
80	**13430925**	**13430925**	0.000	579.04	13432944	0.015	571.96	13431550	0.005	501.32	13432518	0.012	578.78
90	**15109072**	15112686	0.024	380.58	15112848	0.025	541.26	15111091	0.013	572.21	**15109072**	0.000	598.70
100	**16783097**	16784132	0.006	572.83	16784628	0.009	583.78	**16783097**	0.000	454.15	16784365	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

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

No.		GVNS-R			GVNS-G			Adaptive-GVNS-R			Adaptive-GVNS-G		
Units	Best cost	Cost	dev. (%)	Time(s)	Cost	dev. (%)	Time(s)	Cost	dev. (%)	Time(s)	Cost	dev. (%)	Time(s)
10	**3946841**	**3946841**	0.000	11.48	**3946841**	0.000	15.49	**3946841**	0.000	21.54	**3946841**	0.000	21.17
20	**7855658**	7855919	0.003	569.06	**7855658**	0.000	47.39	**7855658**	0.000	172.42	**7855658**	0.000	109.78
30	**11766439**	11766831	0.003	284.92	**11766439**	0.000	316.49	**11766439**	0.000	286.86	**11766439**	0.000	248.36
40	**15684448**	15687211	0.018	582.02	**15684448**	0.000	580.23	15687820	0.022	558.08	15685112	0.004	282.89
50	**19583473**	**19583473**	0.000	575.00	19584708	0.006	509.11	19583700	0.001	565.95	19584330	0.004	599.13
60	**23504315**	**23504315**	0.000	482.46	23506263	0.008	582.30	23505281	0.004	497.49	23505475	0.005	592.99
70	**27426852**	**27426852**	0.000	598.71	27432842	0.022	586.98	27428753	0.007	594.64	27429965	0.011	583.27
80	**31340087**	**31340087**	0.000	597.70	31341845	0.006	587.79	31340474	0.001	591.40	31341820	0.006	593.78
90	**35259360**	**35259360**	0.000	590.48	35261510	0.006	587.69	35262007	0.008	591.80	35260891	0.004	537.39
100	**39162034**	39163354	0.003	598.53	39170260	0.021	592.91	**39162034**	0.000	593.68	39167165	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

From the results presented in Table 7 and Table 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 instance with 10 units and time horizon of 72h, `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.
- On instances with planing horizon of 72h `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 168h, 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 72h. On the other hand, on the instances with planing horizon of 168h, 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.

5 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 UCP, but not VNS. We propose an adaptive mechanism within GVNS which helps to decide what neighborhood structure to apply in some stage of solution process. The computational results show that the proposed VNS methods, with and without the adaptive mechanism, prove to be very efficient in solving UCP, regarding both CPU times spent and result qualities. 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 case considered before.

Proposed methods have been tested in solving general convex UCP problem. 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 UCP and related UCPs with additional constraints such as ramp rate constraints, environmental constraints, emission constraints and so on.

6 Appendix

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	10000	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	64782	796.9	0.3133	220	550	805000	18	8
2	64782	796.9	0.3133	220	550	805000	18	8
3	64670	795.5	0.3127	200	500	805000	18	8
4	64670	795.5	0.3127	200	500	805000	18	8
5	64670	795.5	0.3127	200	500	805000	18	8
6	64670	795.5	0.3127	200	500	805000	18	8
7	64670	795.5	0.3127	200	500	805000	18	8
8	64670	795.5	0.3127	200	500	805000	18	8
9	172832	915.7	0.7075	200	500	402500	7	7
10	172832	915.7	0.7075	114	500	402500	7	7
11	176003	884.2	0.7515	114	500	402500	7	7
12	173028	884.2	0.7083	114	500	402500	7	7
13	91340	1250.1	0.4211	110	500	575000	9	8
14	63440	1298.6	0.5145	90	365	575000	12	8
15	65468	1298.6	0.5691	82	365	575000	12	8
16	72282	1290.8	0.5691	120	325	575000	10	8
17	190928	238.1	25.881	65	315	23000	1	1
18	285372	1149.5	38.734	65	315	23000	1	1
19	271376	1269.1	36.842	65	315	23000	1	1
20	39197	696.1	0.4921	120	272	575000	9	8
21	45576	660.2	0.5728	120	272	575000	9	8
22	28770	803.2	0.3572	110	260	460000	11	8
23	36902	818.2	0.9415	80	190	92000	14	7
24	105510	33.5	52.123	10	150	23000	1	1
25	22233	805.4	11.421	60	125	115000	8	8
26	30953	707.1	20.275	55	110	287500	14	7
27	17044	833.6	30.744	35	75	253000	14	7
28	81079	2188.7	16.765	20	70	5750	1	1
29	124767	1024.4	26.355	20	70	5750	1	1
30	121915	837.1	30.575	20	70	5750	1	1
31	120780	1305.2	25.098	20	70	5750	1	1
32	104441	716.6	33.722	20	60	7670	1	1
33	83224	1633.9	23.915	25	60	7670	1	1
34	111281	969.5	32.562	18	60	7670	1	1
35	64142	2625.8	18.362	8	60	7670	1	1
36	103519	1633.9	23.915	25	60	7670	1	1
37	13547	694.7	8.482	20	38	69000	11	8
38	13518	655.9	9.693	20	38	69000	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

References

[1] Burns, R.M., & Gibson, C.A. (1975). Optimization of priority list for a unit commitment problem. IEEE PES Summer Meeting, 75, 453–461.

[2] Chandram, K., Subrahmanyam, N., & Sydulu, M. (2011). Unit Commitment by improved pre-prepared power demand table and Muller method. International Journal of Electrical Power & Energy Systems, 33, 106–114.

[3] Cheng, C.P., Liu, C.W., & Liu, C.C. (2000). Unit commitment by Lagrangian relaxation and genetic algorithm. IEEE Transactions on Power Systems, 15, 707–714.

[4] Chung, C., Yu, H., & Wong, K.P. (2011). An advanced quantum-inspired evolutionary algorithm for unit commitment. IEEE Transactions on Power Systems, 26, 847–854.

[5] Damousis, I.G.S, Bakirtziz, A.G, & Dokopoulos, P.S. (2004). A Solution to the Unit Commitment problem using integer coded genetic algorithm. IEEE Transactions on Power Systems, 19, 1165–1172.

[6] Dieu, V., & Ongsakul, W. (2006). Enhanced augmented Lagrangian hopfield network for unit commitment. IEE Proceedings on Generation, Transmission and Distribution, 153, 624–632.

[7] Dieu, V., & Ongsakul, W. (2006). Enhanced merit order and augmented Lagrangian hopfield network for ramp rate constrained unit commitment. In: Proceedings of IEEE Power System Society Meeting, Canada.

[8] Dieu, V., & Ongsakul, W. (2011). Augmented Lagrange hopfield network based Lagrangian relaxation for unit commitment. International Journal of Electrical Power & Energy Systems, 33, 522–530.

[9] Dillon, J.D., Walsh, M.P., & O'Malley, M.J. (2002). Initialization of the augmented Hopfield network for improved generator scheduling. IEE Proceedings on Generation, Transmission and Distribution, 149, 593–599.

[10] Ebrahimi, J., Hosseinian, S., & Gharehpetian, G. (2011). Unit commitment problem solution using shuffled frog leaping algorithm. IEEE Transactions on Power Systems, 26, 573–581.

[11] El-Saadawi, M.M., Tantawi, M.A., & Tawfik. E. (2004). A fuzzy optimization-based approach to large scale thermal unit commitment. Electric Power Systems Research, 72, 245–252.

[12] Feo, T.A., & Resende, M.G. (1995). Greedy randomized adaptive search procedures. Journal of Global Optimization, 6, 109–133.

[13] Hansen, P., & Mladenović, N. (1996). Simultaneous static unit commitment and economic dispatch by dynamic programming. Les Cahiers du GERAD, G–96–26, HEC Montréal.

[14] Hansen, P., & Mladenović, N. (2002). A separable approximation dynamic programming algorithm for economic dispatch with transmission losses. Yugoslav Journal of Operations Research, 12, 157–166.

[15] Hansen, P., & Mladenović, N. (2002). Developments of variable neighborhood search (pp. 415–439). Springer US.

[16] Hansen, P., Mladenović, N., & Moreno-Pérez, J.A. (2010). Variable neighbourhood search: methods and applications (invited survey). Annals of Operations Research, 175, 367–407.

[17] Huang, K.Y., Yang, H.T., & Huang, C.L. (1998). A new thermal unit commitment approach using constraint logic programming. IEEE Transactions on Power Systems, 13, 936–945.

[18] Hu, B & Raidl, G. (2006). Variable neighborhood descent with self-adaptive neighborhood-ordering. In Cotta, C., Fernandez, A.J. & Gallardo, J.E. editors, Proceedings of the 7th EU/Meeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics, Malaga, Spain.

[19] Jabr, R.A. (2013) Rank-constrained semidefinite program for unit commitment. International Journal of Electrical Power & Energy Systems, 47, 13–20.

[20] Jeong, Y.W., Park, J.B., Jang, S.H., & Lee, K. (2010). A new quantum inspired binary pso: Application to unit commitment problems for power systems IEEE Transactions on Power Systems, 25, 1486–1495.

[21] Juste, K.A., Kita, H., Tanaka, E., & Hasegawa, J. (1999). An evolutionary programming solution to the unit commitment problem. IEEE Transactions on Power Systems, 14, 1452–1459.

[22] Kazarlis, A., Bakirtzis, A.G, & Petridis, V. (1996). A genetic algorithm solution to the unit commitment problem. IEEE Transactions on Power Systems, 11, 83–92.

[23] Kritzinger, S., Doerner, K.F., Tricoire, F., & Hartl, R.F. (2015). Adaptive search techniques for problems in vehicle routing, Part I: A survey. Yugoslav Journal of Operations Research, 25(1), 3–31, doi:10.2298/YJOR140217009K.

[24] Kritzinger, S., Doerner, K.F., Tricoire, F., & Hartl, R.F. (2015). Adaptive search techniques for problems in vehicle routing, Part II: A numerical comparison. Yugoslav Journal of Operations Research, 25(2), 169–184, doi:10.2298/YJOR140217011K.

[25] Lau, T., Chung, C., Wong, K., Chung, T., & Ho, S. (2009). Quantum-inspired evolutionary algorithm approach for unit commitment. IEEE Transactions on Power Systems, 24, 1503–1512.

[26] Lazić, J., Todosijević, R., Hanafi, S., & Mladenović, N. (2015). Variable and single neighbourhood diving for MIP feasibility. Yugoslav Journal of Operations Research, doi:10.2298/YJOR140417027L.

[27] Lowery, P.G. (1966). Generating unit commitment by dynamic programming. IEEE Transactions on Power Systems, 85, 422–426.

[28] Marcovecchio, M.G., Novais, A.Q., & Grossmann, I.E. (2011). A Branch and Bound Search for the Deterministic Optimization of the Thermal Unit Commitment Problem. Part I: Methodology. http://egon.cheme.cmu.edu/Papers/Marcovecchio-Novais-Grossmann-PartI.pdf.

[29] Marcovecchio, M.G., Novais, A.Q., & Grossmann, I.E. (2011). A Branch and Bound Search for the Deterministic Optimization of the Thermal Unit Commitment Problem. Part II:Computational Results. http://egon.cheme.cmu.edu/Papers/Marcovecchio-Novais-Grossmann-PartII.pdf.

[30] Mhanna, S.N., & Jabr, R.A. (2012). Application of semidefinite programming relaxation and selective pruning to the unit commitment problem. Electric Power Systems Research, 90, 85–92

[31] Mladenović, N., & Hansen, P. (1997). Variable neighborhood search, Computers and Operations Research, 24, 1097–1100.

[32] Mladenović, N., Todosijević, R., & Urošević, D. (2012). An efficient General variable neighborhood search for large TSP problem with time windows, Yugoslav Journal of Operations Research, 22, 141–151.

[33] Moghimi Hadji, M., & Vahidi, B. (2012). A solution to the unit commitment problem using imperialistic competition algorithm. IEEE Transactions on Power Systems, 27, 117–124.

[34] Ongsakul, W., & Petcharaks, N. (2004). Unit commitment by enhanced adaptive Lagrangian relaxation. IEEE Transactions on Power Systems, 19, 620–628.

[35] Pang, C.K., Sheble, G.B., & Albu, F. (1981). Evaluation of dynamic programming based methods and multiple area representation for thermal unit commitment. IEEE Transactions on Power Apparatus and Systems, 100 1212–1218.

[36] Polacek, M., Benkner, S., Doerner, K.F., & Hartl, R.F. (2008). A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. BuR-Business Research, 1(2), 207–218.

[37] Quan, R., Jian, J.,& Mu Y. (2014). Tighter relaxation method for unit commitment based on second-order cone programming and valid inequalities. International Journal of Electrical Power & Energy Systems, 55 82–90.

[38] Rajan, C.C.A., & Mohan, M.R. (2004). An evolutionary programming based Tabu search method for solving the unit commitment problem. IEEE Transactions on Power Systems, 19, 577–585.

[39] Ripon, K.S.N., Glette, K., Khan, K.N., Hovin, M., & Torresen, J. (2013). Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities. Swarm and Evolutionary Computation, 8, 1–12.

[40] Roy, K. (2013). Solution of unit commitment problem using gravitational search algorithm. International Journal of Electrical Power & Energy Systems, 53, 85–94.

[41] Saber, A.Y., Alshareef, A.M. (2008). Scalable unit commitment by memory-bounded ant colony optimization with A* local search. International Journal of Electrical Power & Energy Systems, 30, 403–414.

[42] Saber, A.Y., Senjyu, T., Miyagi, T., Urasaki, N., & Funabashi, T. (2006). Fuzzy unit commitment scheduling using absolutely stochastic simulated annealing. IEEE Transactions on Power Systems, 21, 955–964.

[43] Saber, A.Y., Senjyu, T., Miyagi, T., Urasakiand, N., & Funabashi, T. (2007). Unit commitment by heuristics and absolutely stochastic simulated annealing. IET Generation, Transmission & Distribution, 1, 234–243.

[44] Saber, A.Y., Senjyu, T., Yona, A., Urasaki, N., & Funabashi, T. (2007). Fuzzy unit commitment solution – a novel twofold simulated annealing approach. Electric Power Systems Research, 77, 1699–1712.

[45] Saber, A.Y., Senjyu, T., Yona, A., & Funabashi, T. (2007). Unit commitment computation by fuzzy adaptive particle swarm optimisation. IET Generation, Transmission & Distribution, 1, 456–465.

[46] Sasaki, H., Watanabe, M., & Yokoyama, R. (1992). A solution method of unit commitment by artificial neural networks. IEEE Transactions on Power Systems, 7, 974–981.

[47] Senjyu, T., Yamashiro, H., Shimabukuro, K., Uezato, K., & Funabashi, T. (2002). A unit commitment problem by using genetic algorithm based on characteristic classification. Power Engineering Society, IEEE Winter Meeting, 1, 58–63.

[48] Silva, I. Jr., Carneiro S. Jr., De Oliveira, E.J., Pereira, J.L.R., Garcia P.A.N., & Marcato, A.L.M. (2008). A Lagrangian multiplier based sensitive index to determine the unit commitment of thermal units. International Journal of Electrical Power & Energy Systems, 30, 504–510.

[49] Srinivasan, D., & Chazelas, J. (2004). A priority list based evolutionary algorithm to solve large scale unit commitment problem. In: International Conference on Power System Technology – Powercon 2004, Singapore, 21–24.

[50] Stenger, A., Vigo, D., Enz, S., & Schwind, M. (2013). An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. Transportation Science, 47, 64–80.

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

[52] Swarup K.S., & Yamashiro, S. (2002). Unit commitment solution methodology using genetic algorithm. IEEE Transactions on Power Systems, 17, 87–91.

[53] Todosijević, R., Mladenović, M., Hanafi, S., & Crévits, I. (2012). VNS based heuristic for solving the Unit Commitment problem. Electronic Notes in Discrete Mathematics, 39, 153–160.

[54] Todosijević, R., Mjirda, A, Hanafi, S., & Mladenović, N. (2014). Deterministic use of several neighborhoods in local search: An empirical study on traveling salesman problem, ITQM 2014, Moscow.

[55] Viana, A., & Pedroso J.P. (2013). A new MILP-based approach for Unit Commitment in power production planning. International Journal of Electrical Power & Energy Systems, 44, 997–1005.

[56] Zhao, B., Guo, C.X., Bai, B.R., & Cao, Y.J. (2006). An improved particle swarm optimization algorithm for unit commitment. International Journal of Electrical Power & Energy Systems, 44, 432–512.

[57] Wood, A.J., & Wollenberg, B.F. (1996). Power Generation, Operation and Control, 2nd revised edition. Wiley, New York.