Croatian Operational Research Review
CRORR **9**(2018), 211–221

Dynamic Programming for an Optimal and Equitable Public Load Shedding Schedule

Mohamed Kampo[1], **Babacar Mbaye Ndiaye**[2,*] **and Guy Degla**[1]

[1] *Institute of Mathematics and Physical Sciences, University of Abomey Calavi, BP 613, Porto-Novo, Benin*

E-mail: ⟨mohamed.kampo@yahoo.fr, gdegla@imsp-uac.org ⟩

[2] *Laboratory of Mathematics of Decision and Numerical Analysis, University of Cheikh Anta Diop, BP 45087, Dakar, Senegal*

E-mail: ⟨babacarm.ndiaye@ucad.edu.sn⟩

Abstract. This paper provides a method for an optimal and equitable schedule of public load shedding on any time interval and any number of sectors. By combining dynamic programming and knapsack techniques, the method gives a schedule that iterates over particular intervals. Simulation results with respect to actual schedules of a local energy company show the potential of this new approach for solving such a general problem that challenges many energy worldwide companies.

Key words: modeling, load shedding, dynamic programming, optimization, knapsack problem

Received: February 22, 2018; accepted: August 22, 2018; available online: December 13, 2018

DOI: 10.17535/crorr.2018.0016

1. Introduction

Electrical energy distribution is a critical issue in several countries (mainly in Africa and Asia), where the production of energy cannot satisfy customer demands. A practical solution adopted by most of these countries is to subdivide customers into sectors, and time into intervals, so that during each interval, some sectors may be cut off due to the lack of energy. The main challenge of such a load shedding process is to find a schedule for energy distribution which is fair to customers in the sense that each sector suffers a roughly equal number of cut-off intervals (daily and/or monthly). The schedule should be compatible with the energy production constraints and it should maximize the total consumption of the produced energy. Additional constraints may come from strategic concerns (such as high priority customers like hospitals, airports and government headquarters), as well as socio-economic considerations (such as financial losses that industrial sectors may incur, or the risk of disorder that residential sectors may face).

In this work, we build a new mathematical model for such a problem and we define a scheduling algorithm by combining dynamic programming and knapsack techniques. The algorithm is tested on scenarios, using data from the Beninese Electricity Company (Société Béninoise d'Energie Electrique)[4]. This company uses predefined monthly and static schedules to manage its load shedding process during critical periods. An example of its partitioning scheme is shown in Figure 1 and is based on the local population density. Now, we rather suggest a flexible and dynamic approach to generate a fair and optimal schedule in advance over any selected time period.

The rest of the paper is organized as follows. Section 2 presents related works and discusses their limitations. In section 3, we present our approach, starting with a mathematical formulation of the problem, and ending by our proposed solution algorithm. Section 4 presents the results obtained by this approach applied to various scenarios. Finally, Section 5 concludes the paper and gives some perspectives.

*Corresponding author.

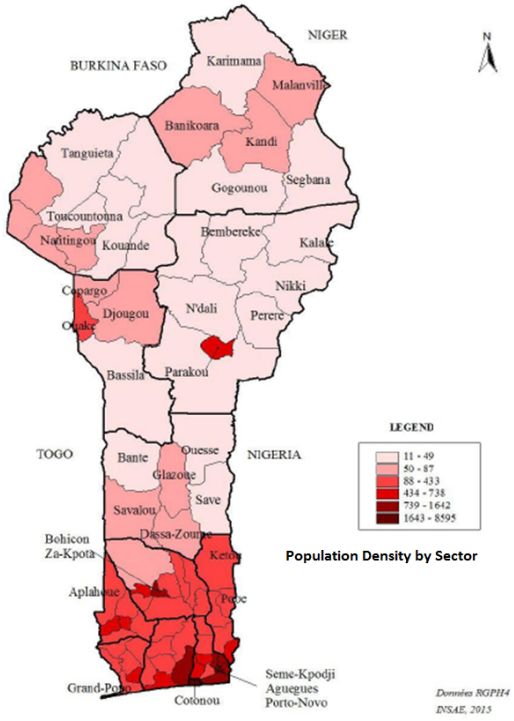

Figure 1: *Example of partitioning scheme in Benin.*

2. Related works

Load shedding is a well-known phenomenon which is required in various situations including but not limited to energy generator failure, transmission cable failures, increase in energy consumption, or decrease of energy production resources. Load alleviation can be obtained by cutting off some customers, either by following some predefined order (e.g., customers are sorted increasingly, decreasingly, or randomly), or according to importance-based priorities and privilege. Obviously, such approaches are not generally optimal although they are used by many energy production and distribution companies.

Recent studies for the causes of blackouts have proved that voltage drops are capable of destabilizing the distribution network. A load shedding algorithm is proposed by Joshi in [11] where frequency and voltage are given as inputs. The disturbance magnitude is estimated using the rate of frequency change and the location. In Isazadeh et al. work [10], an adaptive dynamic load shedding algorithm is proposed to manage an uncontrolled environment. A similar approach is proposed by Kirar et al. in [13], in the context of an industrial cogeneration system. Adaptive load shedding by artificial neural networks is also discussed by Hsu et al. in [6]. Various methods of frequency control are considered by Ameli et al. in [2], with application to operational planning. The authors emphasized the advantages of using the frequency drop gradient parameter and the reasons why it has not been utilized in planning. Other remarkable works can be found in [1], [5] and [16]. All of these efforts have the overall goal of rapidly producing a load shedding plan, which can recover from any instability caused by an incident that has caused the loads to exceed the resources available in a network.

In this paper, we consider a more general situation, where the deficit of energy is not necessarily due to a short-term factor, such as a breakdown, but also to long-term factors such as the lack of means or resources of production to meet the existing demand. Fairness is also a key aspect, in the sense that each sector must totally get the same number of time intervals

at the end of any given cycle. A way to address this concern is to adopt one of the strategies described in Table 1, as done in [7]. In these strategies, H_i are time slots, G_j are groups of sectors. Entry 1 denotes that the indicated group must receive energy in a specific time slot, or 0 otherwise. The number of groups supported at each slot depends on the sum of their load and the number of available resource during this slot. While fairness seems to be theoretically ensured between groups, the practical application of Schedule 1 or Schedule 2 shows that issues remain in case of a variability of sector consumption and/or variability of resource availability. Indeed, such an allocation plan may fail in case the consumption of energy in the early time slots exhausts the number of available resources, therefore making the remaining part of the plan unfeasible. Our approach overcomes this limitation by ensuring a fair and optimal schedule that is compatible with the energy production constraints.

	G_1	G_2	G_3	...	G_p
H_1	0	1	1	...	1
H_2	1	0	1	...	1
H_3	1	1	0	...	1
...	...	...	...	...	...
H_{n-1}	1	1	1	...	1
H_n	1	1	1	...	0

	G_1	G_2	G_3	...	G_p
H_1	1	0	0	...	0
H_2	0	1	0	...	0
H_3	0	0	1	...	0
...	...	...	...	...	...
H_{n-1}	0	0	0	...	0
H_n	0	0	0	...	1

Table 1: *Schedules 1 and 2.*

3. Proposed approach

We propose to model a fair and optimal load shedding program. However, in a first stage, we do not take into account constraints such as priorities given to strategic sectors (like airports, hospitals, and government headquarters). Therefore, all sectors have the same priority. Nevertheless, they do not necessarily get the same number of energy at the end of a cycle since they do not have the same consumption load. The proposed formulation allows to subsequently consider cases where priorities are allocated to specific sectors, as explained later.
Our approach has 2 steps:

1. At the first step (program P_1), we compute the best value of T_s, where T_s denotes the minimum duration during which each sector will receive energy during a cycle.

2. At the second step (program P_2), T_s is used to find the optimal schedule for a cycle.

3.1. Problem formulation

The first-step program (P_1) is defined as follows:

$$Maximize\ T_s$$

$$\begin{cases} \sum\limits_{i \in X_N} B_{it} x_{it} \leq R_t & \forall t \in \zeta_K \qquad (1) \\ \sum\limits_{t \in \zeta_K} x_{it} \geq T_s & \forall i \in X_N \qquad (2) \\ x_{it} \in \{0,1\}, T_s \geq 0 & \forall t \in \zeta_K \ \ \forall i \in X_N \end{cases}$$

where:
i: sector index,

t: time index,
N: number of sectors,
K : number of sequences,
$X_N = \{1, 2, ..., N\}$: set of sectors,
$\zeta_K = \{1, 2, ..., K\}$: set of reference hours,
T_s : minimum number of times each sector receives energy over K hours,
B_{it} : energy requirement of the sector i during the time interval $(t-1, t]$,
R_t : number of resource that must be available during the time interval $(t-1, t]$,
x_{it} : binary variable associated with sector i during the time interval $(t-1, t]$.

Constraint (1) represents the family of knapsack-type constraints ensuring that the total charge of all sectors in a time interval is less than or equal to the number of available energy during this interval. Constraint (2) ensures that the total duration allocated to each sector is greater than or equal to the minimum T_s to ensure per sector.
This first model allows getting an optimal value for T_s, which we will use in the second-step program. The latter (problem P_2) is defined as follows:

$$Maximize \sum_{t\in\zeta_K}\sum_{i\in X_N} B_{it}x_{it}$$

$$\begin{cases} \sum\limits_{i\in X_N} B_{it}x_{it} \leq R_t \quad \forall t \in \zeta_K & (3) \\ \sum\limits_{t\in\zeta_K} x_{it} \geq T_s \qquad \forall i \in X_N & (4) \\ x_{it} \in \{0,1\} \qquad \forall t \in \zeta_K, \forall i \in X_N \end{cases}$$

Constraint (3), respectively (4), is identical to Constraint (1), respectively (2). Therefore, problem P_2 maximizes the total number of energy consumed by all sectors during an entire cycle, while ensuring that all sectors have received energy for at least T_s hours during this cycle, without exceeding the number of available resources.

It is noteworthy that such a formulation allows to easily integrate additional constraints such as priorities allocated to some strategic sectors, or the de-facto allocation of specific durations to specific sectors. In both cases, the x_{it} that correspond to the sectors of concerns are set to 1 before solving the equation. Of course, this must be done without violating the constraints on the resources.

In terms of complexity, the model corresponds to a family of t knapsack-type subprograms, each of which being NP-hard. We use dynamic programming to solve it, taking advantage of the sequential decomposition principle of such an approach to breaking down the program into smaller subprograms and to recursively address them.

3.2. Problem decomposition

Sequential decomposition in dynamic programming retains optimality according to Bellman's principle of optimality [3], which states that: "each sub-policy of an optimum policy must itself be an optimum policy with regard to the initial and the terminal states of the sub-policy".
For each time sequence t,problem P_2 can be rewritten in one of the following two ways (Ascending or Descending):

Ascending ($\mathcal{D}_1$)

$$Maximize \sum_{i\in X_N} B_{it}x_{it}$$

$$\begin{cases} \sum\limits_{i \in X_N} B_{it} x_{it} \leq R_t \\ \\ \sum\limits_{y=1}^{t} x_{iy} \geq T_s + t - K \ \ \forall i \in X_N \\ x_{it} \in \{0,1\} \qquad\qquad \forall i \in X_N. \end{cases}$$

This means that at each time sequence t (taken in an ascending order), each sector must receive at least $T_s - (K - t)$ hours of energy to ensure it will receive at least T_s hours of energy at the end of the cycle.

Descending ($\mathcal{D}_2$)

$$Maximize \quad \sum_{i \in X_N} B_{it} x_{it}$$

$$\begin{cases} \sum\limits_{i \in X_N} B_{it} x_{it} \leq R_t \\ \\ \sum\limits_{y=t}^{K} x_{iy} \geq T_s - t + 1 \ \forall i \in X_N \\ x_{it} \in \{0,1\} \qquad\qquad \forall i \in X_N \end{cases}$$

This means that at each time sequence t (taken in a descending order), each sector must receive at least $T_s - (t-1)$ hours of energy to ensure it will receive at least T_s hours of energy at the end of the cycle. Let $\mathcal{M} = \left\{ i \in X_N \ : \ \sum\limits_{y=1}^{t-1} x_{iy} = T_s + t - K - 1 \right\}$ and $P = \sum\limits_{i \in \mathcal{M}} B_{it}$.

Let us consider the ascending approach. Then, problem P_2 can be reformulated as the following problem ($\mathcal{D}$), which comes down to a conventional Knapsack problem:

$$Maximize \sum_{i \in X_N \setminus \mathcal{M}} B_{it} x_{it}$$

$$\begin{cases} \sum\limits_{i \in X_N \setminus \mathcal{M}} B_{it} x_{it} \leq R_t - P \\ x_{it} \in \{0,1\} \qquad\qquad \forall i \in X_N \end{cases}$$

Many algorithmic solutions are known for such a problem. For the detailed description and related properties of Knapsack problem, we refer the reader to [3, 12, 15].
When all $R_t - P$ (weights) are nonnegative integers, the knapsack problem can be solved in pseudo-polynomial time using dynamic programming. Dynamic programming is well known in the literature and uses a recursive function $F_{(i,c)}$, that helps to find the best candidates for an optimal knapsack problem without having to put them in a specific order according to any criteria. The following algorithm (Algorithm 1) describes the Knapsack approach for solving problem ($\mathcal{D}$).

Notation:
$[0]_{(R_t - P + 1), N}$ is the $(R_t - P + 1) * N$ null matrix
R_t is the t^{th} component of R
X_t is the row t of the matrix X
B_t is the row t of the matrix B.

Algorithm 1: Knapsack algorithm (N,P,X_t,R_t,B_t)

$F = [0]_{(R_t-P+1),N}$
for $i \leftarrow 1$ **to** N **do**
  for $c \leftarrow 0$ **to** $R_t - P$ **do**
    $F_{(0,c)} = 0$
    if $B_{it} \leq c$ **then**
      $F_{(i,c)} = max\{F_{(i-1,c)}; F_{(i-1,c-B_{it})} + B_{it}\}$
    else
      $F_{(i,c)} = F_{(i-1,c)}$
$k = R_t - P$
for $i \leftarrow N$ **to** 1 **do**
  if $F_{(i,k)} = F_{(i-1,k-B_{it})} + B_{it}$ **then**
    $k = k - B_{it}$
    $x_{it} = 1$

3.3. Solution algorithm

Our solution for the entire formalized problem is defined by Algorithm 2, where the program iterates on time sequences. At each step, the algorithm checks for sectors whose needs must be met in priority. In case enough resources remain available to support at least one non-priority sector, the algorithm tries to find the best sub-list of non-priority sectors that can be satisfied (knapsack algorithm).

Algorithm 2: Model (N,K,X,R,B)

for $t \leftarrow 1$ **to** K **do**
  $P = 0$
  $m = 10^6$
  for $i \leftarrow 1$ **to** N **do**
    if $E_i = T_s - K + t - 1$ **then**
      $x_{it} = 1$
      $P = P + B_{it}$
    else if $B_{it} < m$ **then**
      $m = B_{it}$
  if $R_t - P \geq m$ **then**
    Knapsack algorithm
  for $i \leftarrow 1$ **to** N **do**
    $E_i = E_i + x_{it}$

4. Experimental results

Numerical experiments are carried out in order to evaluate the performance of the proposed solution. We generate scenarios in which the number of binary variables is increased by changing the total number of sectors assigned to each sequence. A scenario defines all allocations made for all sectors and at all time sequences. Comparing the performance of our solution to the one adopted in practice by the SBEE company, it appears that our algorithm performs well in a real-world context. We implemente our solution in C++, and all experiments are executed on

a computer system consisting of Intel(R) Core i7, with 8 Gb of RAM, operating under UNIX. We also used CPLEX [8] to solve program P_1, in order to check if the values found by our C++ implementation are optimals.

4.1. Daily and monthly scenarios

We test our algorithm on both daily and monthly planning situations. The results of the computational experiments performed are summarized in Tables 2 and 3, with 6 scenarios for daily tests (numbers of sectors varying from 10 to 200) and 4 scenarios for monthly tests (numbers of sectors varying from 10 to 50). Each scenario is based in real-world data collected from the SBEE company.

Number of sectors	T_s	Total energy consumption
10	15	2244
20	18	5378
25	18	6565
50	18	13298
100	18	26858
200	19	54928

Table 2: *Daily tests.*

Number of sectors	T_s	Total energy consumption
10	513	75220
20	554	161612
25	543	197512
50	547	399242

Table 3: *Monthly tests.*

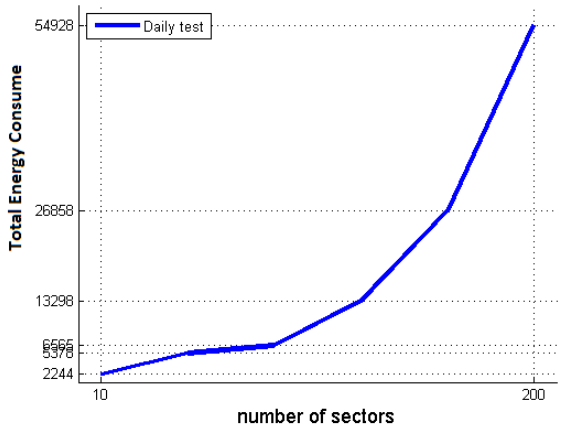

Figure 2: *Daily tests.*

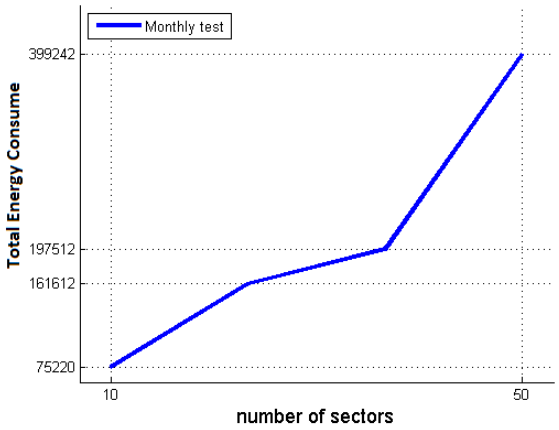

Figure 3: *Monthly tests.*

Figure 2 and Figure 3, indicates the energy evolution for daily tests and the monthly test, respectively.

Since the knapsack algorithm tends to favor sectors at the top of the list, we altere the evaluation order to ensure more equity. This is unfortunately not successful in all cases, since it tends to favor sectors at the middle of the list, although in lesser proportions than what we initially observed.

On the one hand, the lesson we learned here is that our algorithm works well for one-day programs. On the other hand, monthly programs require a better approach (to be adopted) to define priorities in order to maintain equity between sectors until the end of the program.

4.2. Comparison with the SBEE's method

We compare the performance of our algorithm with the SBEE's practical approach, using real-world scenarios of this company. SBEE used to group sectors into four clusters to which a daily schedule allocates two-hour time slots, three clusters at a time (a cluster left one day, is satisfied the next day, while one of the clusters satisfied one day, is left the next day). For our tests, we apply the same principle, i.e., each group undergoes a series of three 2-hour breaks, every day. The model is described by Algorithm 3 hereafter.

Algorithm 3: SBEE Model (N,K,X,s)

$k = 0$
$a = 0$
for $t \leftarrow 1$ **to** K **do**
  $k \leftarrow k + 1$
  for $i \leftarrow 1$ **to** N **do**
    if $((i < a * s)\ ||\ (i >= (1 + a) * s))$ **then**
      $x_{it} = 1$
  if $k\%2 = 0$ **then**
    $a \leftarrow a + 1$
  if $k\%8 = 0$ **then**
    $a = 0$

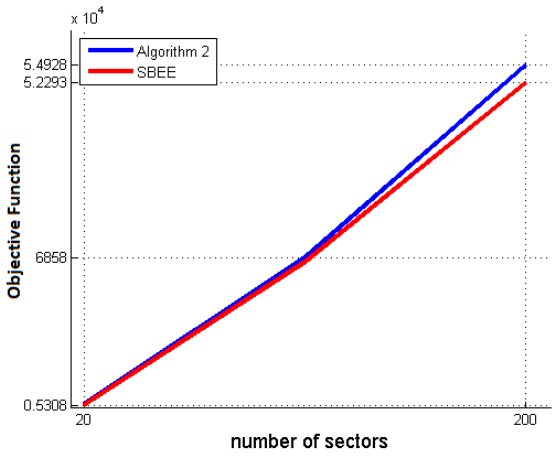

Figure 4: *Comparison between our method and the SBEE approach.*

We perform tests for respective numbers of sectors equal to 20, 100, 200. The results are summarized in Table 4. They show that from our approach, one can provide more energy to each sector in a given time interval, as illustrated by Figure 4. The SBEE's method is more rigid and not suitable for optimizing the objective function. In addition, it happens with this method that at certain hours the available energy is not sufficient to support all three groups, as shown in Table 5 and 6. In practice, when these situations occur, the SBEE agents cut sectors of groups that are needed to be supported.

Number of sectors	Our Algorithm (objective value)	SBEE's method (objective value)
20	5378	5308
100	26858	26156
200	54928	52293

Table 4: *Comparison between our method and the SBEE approach.*

From these results, we see that our algorithm is better than the SBEE approach. In addition, there are many cases where the SBEE approach cannot be rigorously applied because the load will exceed the available resources.

The application of dynamic programming to our model has a considerable impact in terms of complexity and readability of the modeled shedding process. Initially, the model is an integer linear programming problem. It is an NP-hard problem with $N \times K$ variables and $N \times K + N + K$ constraints (including the binary variables constraints). With dynamic programming, we obtain K integer linear subprograms with N variables and $2 \times N + 1$ constraints for each subprogram. This significantly reduces the complexity. On the other hand, the complexity of the dynamic programming algorithm for the knapsack problem is $O(N \times (R_t - P))$ for each of these subprograms.

Hour	Resource	Charge
1	223	214
2	221	225
3	228	223
4	220	214
5	221	219
6	227	233
7	229	224
8	221	226
9	220	231
10	223	227
11	225	213
12	229	214
13	224	227
14	228	221
15	224	207
16	226	204
17	229	234
18	222	214
19	229	224
20	220	225
21	223	208
22	220	230
23	223	220
24	223	231

Table 5: *SBEE's energy consumption evolution per hour*

0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0

Table 6: *Example of SBEE's distribution scheme*

5. Conclusion and perspectives

Load shedding is a traditional solution to the endemic problem of energy scarcity in under-developed countries. In that context, ensuring fairness among customers while optimizing the load shedding plan is a hard task. In this paper, we proposed a general formulation of the problem, and defined a two-step algorithm that finds an optimal schedule.

A limitation of this work is that the energy requirement of a given sector during a given time interval does not depend on what happened in previous time intervals. Therefore, all B_{it} are determined in advance and are static values. This is not always realistic, since in some cases, if a sector does not receive energy during one interval, its customers will try to compensate the loss by more intensive consumption in the next interval. Consequently, B_{it} will rise. In a next work, we will introduce such variations in the demand.

Acknowledgment

This work has been supported by the African Center of Excellence in Mathematical Sciences and Applications (CEA-SMA) project in Benin. The authors would also like to thank Mamadou Kaba Traoré (Clermont Auvergne University), Stanislas Francfort (Orange lab France) and Jules Degila (IMSP, Benin) for their invaluable suggestions. They also thank the anonymous referees for their useful comments and suggestions.

References

[1] Abdullah, A. S. M., Bayejed Bostami Md., Tanvir Yeasin Md. and Hasibur Rahman Md. (2013). An Efficient Load Shedding Scheme from Customer's Perspective. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 2 (10), 5206–5213.

[2] Ameli, M. T., Moslehpour, S. and Rahimikhoshmakani, H. (2006). Presentation and comparison of the various methods of load-shedding for frequency control in Iran power networks. Proceedings of The 2006 IJME- INTERTECH Conference, Kean University, NJ 07083, USA.

[3] Bellman, R. (1954). The theory of dynamic programming. Bulletin of the American Mathematical Society, 60 (6): 503–516, doi:10.1090/S0002-9904-1954-09848-8

[4] Beninese society of electricity (SBEE) company official site (2017). Available from: https://www.sbee.bj/site/

[5] Ghazanfar S., Member, I. and Mahdi E. S. (2012). Effect of Load Shedding Strategy on Interconnected Power Systems Stability When a Blackout Occurs. International Journal of Computer and Electrical Engineering, 4(2), 212–216.

[6] Hsu, C. T., Kang, M. S. and Chen, C. S. (2005). Design of adaptive load shedding by artificial neural networks. IEE Proceedings, 152(3), 415–421. doi:10.1049/ip-gtd:20041207

[7] Husna S., Md Pauzi A., Iqbal F., Mohammad Y. H. and Faridah H. (2016). An Improved Load Shedding Scheduling Strategy for Solving Power Supply deficit. Jurnal Teknologi (Sciences and Engineering), 78(5–7), 61–66. doi: 10.11113/jt.v78.8714

[8] IBM ILOG CPLEX Optimization Studio V12.5.1 (2013). Inc. Using the CPLEX^R Callable Library and CPLEX Barrier and Mixed Integer Solver Options. Available at: http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio

[9] Isazadeh, G., Hooshmand, R. A. and Khodabakhshian, A. (2011). Modeling and optimization of an adaptive dynamic load shedding using the ANFIS-PSO algorithm. Simulation: Transactions of the Society for Modeling and Simulation International, 88(2), 181–196. doi: 10.1177/0037549711400452

[10] Isazadeh, G. H., Hooshmand, R. and Khodabakhshian, A. (2012). Design of an Adaptive Dynamic Load Shedding Algorithm Using Neural Network in The Steelmaking Cogeneration Facility. Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 36(E1), 67–82.

[11] Joshi, P. M. (2007). Load Shedding Algorithm Using Voltage and Frequency Data. All Theses. Paper 240. Clemson University.

[12] Kellerer, H., Pferschy, U. and Pisinger, D. (2004). Knapsack Problems. Springer. doi:10.1007/978-3-540-24777-7

[13] Kirar, M. K., Kamdar, R., Kumar, M. and Agihotri, G. (2013). Load shedding design for an industrial cogeneration system. Electrical and Electronics Engineering: An International Journal, 2(2), 35–46.

[14] Lu W. (2009). Le délestage optimal pour la prévention des grandes pannes d'électricité. Mémoire de Thèse Institut Polytechnique de Grenoble. https://tel.archives-ouvertes.fr/tel-00405654/document [Accessed 22/2/2018]

[15] Martello, S. and Toth, P. (1990). Knapsack problems: Algorithms and computer implementations. Wiley, New York. Available at www.or.deis.unibo.it/knapsack.html

[16] Xu, H., Topcu, U., Low, S. H., Clarke, C. R. and Chandy, K. M. (2010). Load-shedding probabilities with hybrid renewable power generation and energy storage. 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Allerton, IL, 233–239. doi: 10.1109/ALLERTON.2010.5706912