Vectorized Lambda Iteration Method for Swift Economic Dispatch Analysis

Rifki Rahman Nur Ikhsan
School of Electrical Engineering, Telkom University

Raharjo, Jangkung
School of Electrical Engineering, Telkom University

Rahmat, Basuki
School of Electrical Engineering, Telkom University

https://doi.org/10.5109/7172306

出版情報：Evergreen. 11 (1), pp.435-447, 2024-03. 九州大学グリーンテクノロジー研究教育センター
バージョン：

Vectorized Lambda Iteration Method for Swift Economic Dispatch Analysis

Rifki Rahman Nur Ikhsan[1], Jangkung Raharjo[1], Basuki Rahmat[1]
[1]School of Electrical Engineering, Telkom University, Indonesia

*Author to whom correspondence should be addressed:
E-mail: rifkirah@gmail.com

(Received October 19, 2023; Revised January 27, 2024; Accepted January 31, 2024).

Abstract: The economic dispatch process in power markets aims to find the optimal generation schedule based on cost-based bids from generators. The lambda iteration method is commonly used but faces computational challenges for large-scale systems. To address this, a novel approach called the vectorized lambda iteration method is proposed. This method utilizes vectorization techniques to reduce computation time while maintaining accuracy. By parallelizing the computation of schedules and Lagrange multipliers, it achieves faster convergence and enables real-time decision-making. The method improves efficiency, competitiveness, and adaptability in power markets, ensuring reliable operation. Software and hardware optimizations further enhance performance. The results demonstrate a significant reduction in computation time and the lowest cost across three test systems. The vectorized lambda iteration method offers an efficient solution for power system operation and decision-making, with the potential for optimization in practical applications. The proposed vectorized variant that we introduced is capable of reducing computation time by 99% compared to conventional methods, while also achieving a 2% cost reduction.

Keywords: economic dispatch; lambda iteration method; python; vectorization technique

1. Introduction

In the electrical system, we often encounter economic dispatch (ED), optimal power flow (OPF), and security-constrained optimal power flow (SCOPF) issues. A comparison between these three is presented by Obio and colleagues[1)].

In power markets, the economic dispatch process typically involves power generators submitting their cost-based bids for each hour[2)]. These bids are used by the system operator to determine the optimal generation schedule that meets the demand at the lowest possible cost. However, the time required for the economic dispatch (ED) computation can be a bottleneck, especially when dealing with large-scale power systems and complex market structures.

The term ED refers to the optimal and economical distribution of loads among the existing generating units in the system at a specific load price[3)]. Several methods for ED have been published. The Lagrange multiplier method is applied to Economic Dispatch (ED) in the thermal power generation system of Jeneponto, Indonesia[4)]. The Large to Small Area Technique (LSAT) and the Technique of Narrowing Down Area (TONDA) are applied in Economic Dispatch (ED) with the principle of reducing the feasible area. In LSAT[5)], the feasible area is formed from the overall constraints of the generators. A population of candidate solutions is scattered within this feasible area. After finding the best candidate, the dimension of the feasible area is reduced, and the same number of candidates is scattered within it as before. This process continues until a final solution is obtained. Meanwhile, in TONDA[6)], the methodology involves narrowing down the Power Dispatch Generation Limits (PDGL) of each generator. In each iteration, PDGL is divided into segments, and each segment boundary is represented as a solution point. Once the best candidate is obtained, PDGL is further narrowed to improve that candidate. The iteration process continues to update PDGL (after being narrowed down from the previous iteration) until the smallest segment is reached. The smallest area is formed within the narrowest PDGL, where this smallest area can be expressed as a solution point (convergence point).

A new method applied in ED is the Komodo Mlipir Algorithm (KMA). This algorithm adopts the hunting and breeding behavior of Komodo dragons. KMA divides the candidate solutions into three groups based on Komodo individuals: big males, females, and small males. The KMA algorithm begins with the movement of the big male, referred to as high-exploitation low exploration (HILE), followed by females attempting to find solutions by mating with the best big male (highest quality) or through parthenogenesis (exploration)[7)].

Abttan and colleagues published the Ant Lion Optimization Algorithm (ALOA) and Bat Algorithm (BA)

methods for Economic Dispatch (ED) and compared them with various methods[8]. Meanwhile, Xiong and colleagues implemented Across Neighborhood Search (ANS) for ED. In ANS, a group of individuals collaboratively navigates the search space to obtain an optimal solution while simultaneously exploring the neighborhoods of several superior solutions[9]. The application of Particle Swarm Optimization (PSO), Termite Colony Optimization (TCO), and Cat Swarm Optimization (CSO) to solve ED problems was published by [10,11]. Several other algorithms for solving ED problems have also been published, such as the Adaptive Backtracking Search Optimization Algorithm[12], Modified Whales Optimization Algorithm[13], A Novel Hybrid Moth Flame Optimization with Sequential [14], the CSAJAYA algorithm[15], Genetic Algorithm Fuzzy Approach[16,17], and Manta Ray Foraging Optimization technique[18].

The lambda iteration method is a well-established technique for solving the economic dispatch problem. It involves iteratively adjusting the Lagrange multipliers, known as lambdas until the equilibrium conditions are met. This iterative process allows for the optimization of generation schedules while considering various operational constraints. However, the lambda iteration method can be computationally intensive, especially when applied to large-scale systems with numerous generating units[19].

To address the computational challenges associated with the lambda iteration method, this paper proposes a novel approach called the vectorized lambda iteration method. The main idea behind this method is to leverage vectorization techniques in programming to perform computations on multiple elements simultaneously. By exploiting the inherent parallelism in the economic dispatch problem[20], the proposed method aims to significantly reduce the computation time while maintaining a high level of accuracy.

Vectorized programming techniques enable the simultaneous execution of mathematical operations on arrays or vectors of data[21]. By utilizing optimized algorithms and utilizing parallel processing capabilities of modern computer architectures, the vectorized lambda iteration method can perform the necessary calculations more efficiently. This approach allows for faster convergence and reduces the overall computational burden[22], making it well-suited for real-time bidding scenarios.

The proposed method builds upon the foundation of the lambda iteration method but introduces significant computational enhancements through vectorization. By parallelizing the computation of generation schedules and the corresponding Lagrange multipliers, the vectorized lambda iteration method can achieve a substantial reduction in the time required to obtain the optimal dispatch solution[22]. The reduction in computation time provided by the vectorized lambda iteration method has several advantages. Firstly, it enables faster bidding decisions by generators, allowing them to adapt to changing market conditions more promptly. This can enhance the overall efficiency and competitiveness of the power market. Secondly, the reduced computation time facilitates real-time decision-making, which is crucial for managing fluctuations in electricity demand and optimizing the utilization of generation resources.

The vectorized lambda iteration method not only offers improved computational efficiency but also preserves the accuracy of the economic dispatch solution. By incorporating advanced vectorized programming techniques, the proposed method can handle complex system constraints and accurately determine the optimal dispatch schedule, ensuring reliable and stable power system operation. The implementation of the vectorized lambda iteration method requires careful consideration of software and hardware optimization. Utilizing specialized libraries, such as NumPy or TensorFlow, can provide efficient and scalable vectorization capabilities[20]. Furthermore, hardware acceleration techniques, such as utilizing graphics processing units (GPUs) or field-programmable gate arrays (FPGAs), can further enhance the performance of the method.

In conclusion, this paper presents the vectorized lambda iteration method as a novel approach to address the computational challenges associated with economic dispatch bidding. By leveraging vectorization techniques, the proposed method offers a significant reduction in computation time while maintaining high accuracy [20]. The time efficiency and accuracy provided by the vectorized lambda iteration method can lead to improved decision-making and enhance the overall operation of power systems in dynamic market environments. Future research can explore further optimization and refinement of the method and evaluate its performance in practical power system applications.

2. Basic Concepts

2.1. Economic Dispatch

When it comes to managing the electric power system, economic dispatch poses a challenging optimization issue. The objective is to distribute the power generated by various sources in a way that minimizes the cost of producing electricity. While the economic dispatch problem can be viewed as a linear programming challenge, finding the optimal solution is not a straightforward task due to the intricate nature of the variables involved. To address this problem, specific algorithms like Lambda Iteration Methods, Gradient Methods, or artificial methods such as Multi Dimension of Coarse to Fine Search[23], Large to Small Area Technique[5], Komodo Mlipir Algorithm[7], Cuckoo Algorithm, Whale Optimization Algorithm[24], Particle Swarm Optimization[25], Cat Swarm Optimization[25], Ant Colony Optimization[25], and Artificial Bee Colony Algorithm[25].

The main goal of an economic dispatch problem is to

minimize the overall expenses associated with fuel consumption, while also considering the limitations and restrictions of a power generation system. This objective is typically expressed mathematically in the form of an equation (1).

$$Minimize: F_T = \sum_{i=1}^{n} F_i(P_i) \quad (1)$$

Where

F_T : Total generation cost
F_i : Generation cost function
P_i : Power production
i : Generator

In the context of power system operation, the total generation cost F_T is a crucial factor that depends on several variables. One of these variables is n, which represents the number of generators committed to the operating system. Each generator's contribution to the total cost is determined by its generation cost function F_i. The generation cost function for the i th generator can be mathematically expressed as a quadratic polynomial, as depicted in equation (2). This quadratic representation allows for modeling the relationship between the generator's output and its corresponding cost in a more accurately and practically manner. By considering these factors and employing quadratic cost functions, power system operators can make informed decisions regarding the optimal commitment of generators to minimize the total generation cost while meeting the electricity demand efficiently[26]. This mathematical framework forms the basis for various optimization techniques used in power system economics and planning.

$$F_i(P_i) = a_i + b_i P_i + c_i {P_i}^2 \quad (2)$$

Where

F_i : Generation cost function
P_i : Power production
$a_i,\ b_i,\ c_i$: Cost coefficients

In the context of power generation, the variables a_i, b_i, and c_i represent the cost coefficients of the i th generator, while P_i represents the power production of the i th generator. These cost coefficients play a crucial role in determining the operational cost of each generator, and they are typically used in economic dispatch and optimization models for power systems.

2.1.1. Power Balance Constraint (PB)

The total power generated by all units in a power system is balanced with the total power consumed by loads and losses, as represented by equation (3). This balance ensures the equilibrium of the power system.

$$\sum_{i=1}^{n} P_i = P_D + P_{Loss} \quad (3)$$

Where

P_i : Power production
P_D : Load demand
P_{Loss} : Total transmission loss

In the given context, P_D represents the load demand, which signifies the amount of power required, while P_{Loss} indicates the total transmission loss that is intentionally omitted or neglected in the analysis. Neglecting the transmission loss simplifies the calculations and allows a focus on the primary load demand without considering the energy losses incurred during power transmission.

2.1.2. Generation Limits Constraint (GL)

Equation (4) shows the upper and lower limits of the power output for each generator. These limits define the range within which the generator can operate.

$$P_{i,min} \le P_i \le P_{i,max} \quad (4)$$

Where

$P_{i,min}$: Lower limits of the power output
$P_{i,max}$: Upper limits of the power output

In the context of power generation units, the variables $P_{i,min}$ and $P_{i,max}$ denote the minimum and maximum limits of the i th generating unit's power outputs, respectively. These limits play a crucial role in regulating and controlling the power generation process. By setting appropriate lower and upper bounds for each unit, operators can ensure the stability and efficiency of the overall power system. It allows them to prevent excessive power generation that might lead to equipment damage, as well as ensure that the units are operating optimally within their specified capacity.

2.1.3. Ramp Rate Constraint (RR)

The limits on how quickly the power output of a generator can be changed over time as shown in equations (5) and (6).

$$P_{i,t} - P_{i,t-1} \le U_i \quad (5)$$

$$P_{i,t-1} - P_{i,t} \le D_i \quad (6)$$

Where

$P_{i,t}$: Current power demands
$P_{i,t-1}$: Previous power demands
U_i : Ramp Up Rate
D_i : Ramp Down Rate

In the given context, the variables $P_{i,t}$ and $P_{i,t-1}$ represent the power output from the current and previous power demands of the i th generating unit, respectively. D_i and U_i correspond to the lower and higher ramp rates

for the power outputs of the i th generating unit. These ramp rates determine the limits within which the power output can change over time, ensuring the stability and smooth operation of the power generation system. By considering the differences between the current and previous power demands and setting bounds with the ramp rates, the power generation process can be efficiently managed and controlled.

2.2. Lambda Iteration Method

The Lambda iteration method is an efficient and accurate iterative optimization algorithm used for solving constrained optimization problems like economic dispatch. It transforms the problem's constraints into Lagrange multipliers, which act as penalty terms in the objective function. The algorithm iteratively updates the Lagrange multiplier and power generation values to find an optimal solution that minimizes fuel consumption costs while satisfying the power demand and plant constraints.

By incorporating the Lagrange multiplier and constraints, the Lambda iteration method ensures that the power generation plan optimally minimizes fuel costs while meeting the power demand. This iterative approach gradually refines the Lagrange multiplier and power generation values as shown in equations (7) and (8)[27], narrowing the gap between generation and demand while respecting the power plant constraints. This results in an efficient and accurate solution to the economic dispatch problem.

The Lambda iteration method's effectiveness lies in its ability to consider factors such as fuel costs, plant capacities, and operational constraints. By iteratively updating the Lagrange multiplier and power generation values, it provides power system operators with informed decisions for allocating power generation. This robust approach enables efficient optimization of economic dispatch problems, balancing the trade-off between fuel consumption costs and power demand satisfaction while accounting for the constraints of each power plant.

$$L = \sum_{i=1}^{n}\left(a_i + b_i P_i + c_i {P_i}^2\right) - \lambda\left(\sum_{i=1}^{n} P_i - P_D\right) \tag{7}$$

$$P_i = \frac{(\lambda - b_i)}{2c_i} \tag{8}$$

The economic dispatch problem, employing the lambda iteration method, can be resolved by utilizing the equation formulated using the first-order Taylor series expansion. This equation incorporates the variable 'k', which represents an integer denoting the iteration sequence within a numerical method. $\lambda^{(k+1)}$ represents the updated value of lambda in each iteration as shown in equation (9)[27].

$$\lambda^{(k+1)} = \lambda^k + \left(\left(P_D - \sum_{i=1}^{n}\left(\frac{(\lambda^k - b_i)}{2c_i}\right)\right) \Big/ \sum_{i=1}^{n}\frac{1}{2c_i}\right) \tag{9}$$

Where

$\lambda^{(k+1)}$: Updated value

λ : Lagrange multiplier factor

k : Iteration sequence

2.3. Vectorization Technique

Vectorization is a powerful technique in computer science that boosts processor performance by processing multiple data elements concurrently. It leverages specialized instructions designed for vector operations to execute mathematical computations on arrays of data, known as vectors, rather than individual scalar values. This approach significantly improves efficiency when dealing with large arrays, as it reduces the number of instructions required and optimizes memory utilization. The Arithmetic Logic Unit (ALU) is a critical component in vectorization. It performs arithmetic and logical operations on data stored in registers, which are high-speed memory locations within the processor. By utilizing vectorization, the ALU can process multiple data elements simultaneously, leading to enhanced performance and reduced instruction count. This technique maximizes memory bandwidth and cache resources, resulting in improved efficiency when working with large datasets.

These are the mathematical representations of how vectorization is applied in lambda iteration for generator scheduling.

$$P_X = \frac{\left(\begin{bmatrix}\lambda_i \\ \vdots \\ \lambda_n\end{bmatrix} - \begin{bmatrix}b_i \\ \vdots \\ b_n\end{bmatrix}\right)}{2\begin{bmatrix}c_i \\ \vdots \\ c_n\end{bmatrix}} \tag{10}$$

$$\lambda^{(k+1)} = \lambda^k + \left(\left(P_D - \sum_{i=1}^{n}\left(\frac{\left(\lambda^k - \begin{bmatrix}b_i \\ \vdots \\ b_n\end{bmatrix}\right)}{2\begin{bmatrix}c_i \\ \vdots \\ c_n\end{bmatrix}}\right)\right) \Big/ \sum_{i=1}^{n}\frac{1}{2\begin{bmatrix}c_i \\ \vdots \\ c_n\end{bmatrix}}\right) \tag{11}$$

$$F_X(P_X) = \begin{bmatrix}a_i \\ \vdots \\ a_n\end{bmatrix} + \begin{bmatrix}b_i \\ \vdots \\ b_n\end{bmatrix}.\left(\frac{\left(\left(\begin{bmatrix}\lambda_i \\ \vdots \\ \lambda_n\end{bmatrix} - \begin{bmatrix}b_i \\ \vdots \\ b_n\end{bmatrix}\right)\right)}{2\begin{bmatrix}c_i \\ \vdots \\ c_n\end{bmatrix}}\right) + \begin{bmatrix}c_i \\ \vdots \\ c_n\end{bmatrix}.\left(\frac{\left(\left(\begin{bmatrix}\lambda_i \\ \vdots \\ \lambda_n\end{bmatrix} - \begin{bmatrix}b_i \\ \vdots \\ b_n\end{bmatrix}\right)\right)}{2\begin{bmatrix}c_i \\ \vdots \\ c_n\end{bmatrix}}\right)^2 \tag{12}$$

Where

P_X : power production (vectorization)

F_X : Generation cost function (vectorization)

In the context of power generation, F_X represents the total cost amount incurred from all generators, while P_X represents the total power produced by all generators. Equations (10), (11), and (12) are mathematical expressions that implement vector operations, building upon the concepts introduced in equations (2), (8) and (9).

In the field of programming, vectorization is a commonly employed method with the goal of optimizing the execution of operations on complete arrays or vectors, as opposed to handling individual elements separately. This approach leads to substantial enhancements in both computational speed and the overall performance of the code[20]. This approach is particularly valuable when dealing with large datasets and complex calculations, making it a fundamental concept in data science, machine learning, and scientific computing. By leveraging vectorization, programmers can take advantage of specialized hardware and libraries that support parallel processing, leading to faster execution of tasks and overall program efficiency.

Consequently, vectorization plays a crucial role in optimizing algorithms and developing high-performance software solutions. Processors execute vectorized operations with a single instruction, enabling parallel processing of multiple data elements and reducing the instruction count, thus enhancing efficiency and performance. Modern processors further enhance vectorization through features like Single Instruction Multiple Data (SIMD) instructions, which process multiple data elements simultaneously and maximize hardware resource utilization[21].

One of the popular tools for vectorization in Python is the NumPy library. NumPy provides a wide range of functions and tools optimized for handling arrays and matrices, enabling users to perform vectorized operations with ease. It utilizes highly efficient, low-level routines written in C or Fortran, making it an excellent choice for handling large datasets and performing complex mathematical calculations. In contrast to traditional iterative methods that involve explicit loops, vectorization using NumPy offers a more concise and elegant way to express mathematical operations. The code becomes more readable and easier to maintain, promoting better development practices and reducing the chances of introducing errors.

2.4. VLIM's Algorithm and Flowchart

The following is the VLIM algorithm for calculating power production along with its cost in the case of 24 hours power demand scenario.

Step 1 → Import the required libraries (Pandas, NumPy, DataFrame, time).

Step 2 → Read input data from the 'input4.xlsx' and 'power_demand.xlsx' files using Pandas.

- Create a variable 'data' to store the data from 'input4.xlsx'.
- Create variables 'a', 'b', 'c', 'Minimum_Capacity', 'Maximum_Capacity', 'ramp_up', 'ramp_down', and 'Unit' to store the respective columns from 'data'.
- Create a variable 'power_demand_data' to store the data from 'power_demand.xlsx'.
- Create a variable 'power_demand' to store the data from the 'load' column of 'power_demand_data'.

Step 3 → Define the 'calculate_power' function with parameters:

- demand (power demand)
- a (parameter 'a')
- b (parameter 'b')
- c (parameter 'c')
- pmin (minimum capacity)
- pmax (maximum capacity)
- rampup (ramp-up rate)
- rampdown (ramp-down rate)

Step 4 → Define the 'calculate_fuel_cost' function with parameters:

- P (an array of power produced by each unit)
- a (parameter 'a')
- b (parameter 'b')
- c (parameter 'c')

The 'calculate_fuel_cost' function has local variables:

- squaredP (an array containing the squared power values produced)
- Fuel_Cost (an array containing the fuel cost for each unit)
- total_Fuel_Cost (a variable to store the total fuel cost)

In the 'calculate_fuel_cost' function:

- Calculate 'squaredP' as the result of squaring the 'P' array.
- Calculate 'Fuel_Cost' as an array containing the fuel cost for each unit based on the given formula. This part implemented vectorization as shown in equation (12).
- Calculate 'total_Fuel_Cost' as the sum of all fuel costs.
- Return the 'Fuel_Cost' array and the 'total_Fuel_Cost'.

Step 5 → Loop through each power demand value from the 'power_demand' data:

- For each demand value:
- Record the start time of the execution.
- Calculate the power produced ('P') and the number of iterations required using the 'calculate_power' function.
- Calculate the fuel cost ('Fuel_Cost') and the total fuel cost ('total_Fuel_Cost') using the 'calculate_fuel_cost' function.
- Record the end time of the execution.
- Create a DataFrame 'output' to store the calculation results (Unit, Power Produced (MW), Fuel Cost (Rp)).
- Print the calculation results for the current power demand.
- Display a separator line for the next calculation result.

Step 6 → End the process.

As depicted in Fig. 1, the flowchart illustrates the iterative process of VLIM based on the algorithm explained earlier.

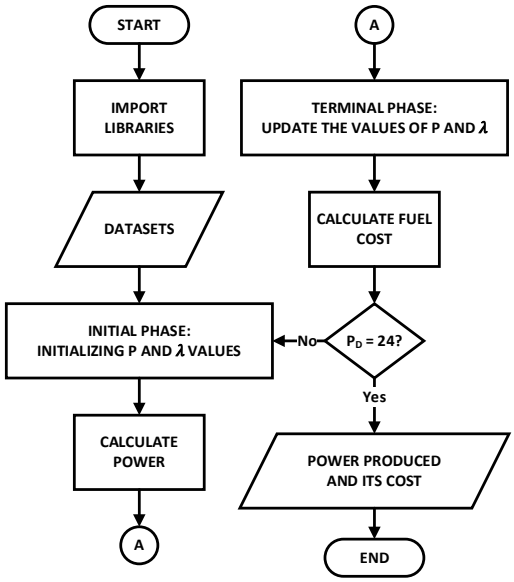

Fig. 1: VLIM's Flowchart

3. Datasets

To evaluate the appropriateness and effectiveness of the vectorized lambda iteration method, a series of three comprehensive tests were conducted. The datasets utilized in these tests form a crucial component of the assessment process. These datasets were specifically chosen to rigorously examine the performance and reliability of the vectorized lambda iteration method under various conditions and scenarios.

3.1. Datasets of Test System 1

During the first test, a configuration employing a 15-unit generator was implemented to meet a substantial load demand of 2,650 MW, as explicitly outlined in both Table 1 and Table 2.

a. Fuel Cost Coefficient

Table 1. Fuel Cost Coefficient of Test System 1

Unit	a	b	c
	(IDR)	(IDR/MW)	(IDR/MW^2)
1	671.03	10.1	0.000299
2	574.54	10.22	0.000183
3	374.59	8.8	0.001126
4	461.37	8.8	0.001126
5	630.14	10.4	0.000205
6	1.661	10.1	0.000301
7	548.2	9.87	0.000364
8	227.09	11.5	0.000338
9	173.72	11.21	0.000807
10	175.95	10.72	0.001203
11	186.86	11.21	0.003586
12	230.27	9.9	0.005513
13	225.28	13.12	0.000371
14	309.03	12.12	0.001929
15	323.79	12.41	0.004447

b. Constraint (Generation Limits)

Table 2. The Generation Limits of Test System 1

Unit	P_{min}	P_{max}
	(MW)	
1	150	455
2	150	455
3	20	130
4	20	130
5	150	470
6	135	460
7	135	465
8	60	300
9	25	162
10	20	160
11	20	80
12	20	80
13	25	85
14	15	55
15	15	55

3.2. Datasets of Test System 2

In the second testing phase, we utilized an extended configuration with a 16-unit generator to meet the dynamic load demands spanning a 24-hour period, as detailed in Table 3. This comprehensive setup facilitated a meticulous assessment of the system's adaptability and responsiveness to fluctuating load conditions throughout the day. Due to the nature of the research, ethical considerations prohibit the disclosure of specific data pertaining to the characteristics of each generator. The presented data incorporates confidential information sourced from an undisclosed third-party associated with the research.

Table 3. Power Demand of Test System 2

Hour	Power Demand (MW)	Hour	Power Demand (MW)
1	4,280.712	13	4,277.938
2	4,256.056	14	4,331.556
3	4,206.554	15	4,517.064
4	4,186.648	16	4,635.494
5	4,154.374	17	4,824.438
6	4,131.04	18	4,911.755
7	4,189.486	19	4,971.81
8	4,163.306	20	5,032.035
9	4,208.538	21	5,097.745
10	4,279.786	22	5,105.55
11	4,338.03	23	5,070.18
12	4,276.804	24	4,915.15

3.3. Datasets of Test System 3

In the third experiment, a 42-unit generator was employed under dynamic load demand conditions (as per the 24-hour demand outlined in Table 4). Given the research's nature, ethical considerations prevent the disclosure of specific data regarding each generator's characteristics. The provided data includes confidential information obtained from an undisclosed source associated with a third party.

Table 4. Power Demand of Test System 3

Hour	Power Demand (MW)	Hour	Power Demand (MW)
1	8,761.424	13	8,876.06
2	8,712.112	14	8,753.608
3	8,613.108	15	8,755.876
4	8,573.296	16	8,863.112
5	8,508.748	17	9,234.128
6	8,462.08	18	9,470.988
7	8,378.972	19	9,848.876
8	8,526.612	20	10,023.51
9	8,617.076	21	10,143.62
10	8,759.572	22	10,264.07
11	8,712.112	23	10,395.49
12	8,613.108	24	10,411.1

4. Experimental Results

In test system 1, only power balance was considered as an accuracy benchmark. The results will be compared with Enhanced Lambda Iteration Method (ELIM) and conventional Lambda Iteration Method (LIM). Test system 2 considered power limits constraint, power balance constraint, and ramp rate. The results will be compared with conventional Lambda Iteration Method (LIM), hybrid Lambda Iteration with Whale Optimization Algorithm (LIM-WOA), hybrid Lambda Iteration with Gravitational Search Algorithm (LIM-GSA), and Artificial Bee Colony (ABC). Test system 3 considered power limits constraint, power balance constraint, and ramp rate. The results will be compared with conventional Lambda Iteration Method (LIM), hybrid Lambda Iteration Method with Whale Optimization Algorithm (LIM-WOA), hybrid Lambda Iteration with Gravitational Search Algorithm (LIM-GSA), Artificial Bee Colony (ABC), and Genetic Algorithm (GA).

4.1. Test System 1

Table 5 provides evidence that the vectorized lambda iteration method exhibits notable advantages over enhanced lambda iteration and conventional lambda iteration in terms of computation time.

Table 5. Test System 1 Results

Method	VLIM	ELIM [28]	LIM [28]
Cost ($/h)	32,183.1587	32,542.4376	32,549.8
Time (sec)	0.00494	0.1233	2.556

It demonstrates a significantly faster computational performance, surpassing enhanced lambda iteration by reducing the computation time by up to 96%. This highlights the efficiency and effectiveness of the vectorized lambda iteration approach in solving optimization problems. Furthermore, the benefits of employing the vectorized lambda iteration method extend beyond computational speed. When considering the total cost aspect, the results indicate that the vectorized lambda iteration approach yields the most cost-effective outcome compared to the other lambda iteration variants. This implies that not only does it offer computational efficiency, but it also generates the most economically favorable solution.

The combination of faster computation time and lower total cost positions the vectorized lambda iteration method as a superior choice for optimization problems, outperforming both enhanced lambda iteration and conventional lambda iteration methods. These findings highlight the practical advantages of utilizing the vectorized lambda iteration approach in real-world scenarios, contributing to improved efficiency and cost savings.

4.2. Test System 2

In Fig. 2, the graph illustrates the relationship between power production (in Giga-watts) and the corresponding demand at different hours. The x-axis represents the scale of power production values, while the y-axis indicates the specific hour of the demand. Upon analyzing the results depicted in the graph, it becomes evident that all algorithms, except LIM-GSA, did not achieve a perfect 100% accuracy in meeting the power balance constraint. This implies that there were instances where the power produced did not precisely match the demand requirements for certain hours. As depicted in Table 6, which delineates the scheduling of power generators during the initial hour load, the recorded amount stands at 4,280.712 MW.

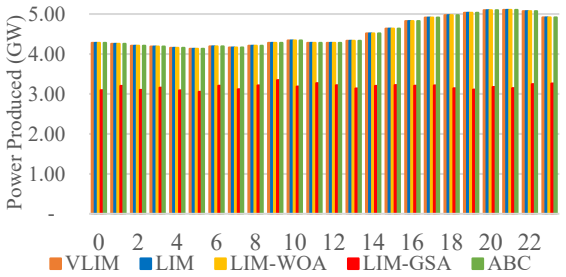

Fig. 2: Test System 2 Result in Term Power Production

Table 6. Power Allocation at 4,280.712 MW

Unit	VLIM	LIM-WOA	LIM-GSA	ABC
1	371.5	371.5	250	300.5939
2	555.7512	510.4489	431.6861	509.5317
3	174.6	174.6	112.6197	184.9037
4	80	114.6	40	75.68308
5	170	170	75	97.57115
6	451.356	414.5635	220	326.9291
7	16	294.5635	5	15.24281
8	118.5	118.5	60	129.7948
9	458.6713	298.5	400	475.8588
10	140	140	120	153.1352
11	326.127	299.5426	200	353.4714
12	367.5788	345	345	598.6724
13	266.6277	244.8934	203.435	257.7388
14	200	200	200	291.4143
15	214	214	180	212.843
16	370	370	250	297.3278
Total	4280.712	4280.712	3092.741	4280.712

However, it is important to note that despite not reaching complete accuracy, several algorithms still managed to satisfy the power balance constraint to a significant extent. Notably, the VLIM, conventional LIM, LIM-WOA, and ABC algorithms demonstrated their capability to fulfill the power balance requirement within an acceptable range. Although these algorithms did not achieve absolute accuracy, they effectively ensured that the power produced was closely aligned with the demand, minimizing significant deviations or imbalances. This indicates their effectiveness in maintaining a relatively stable power generation system, where the production closely corresponds to the prevailing demand.

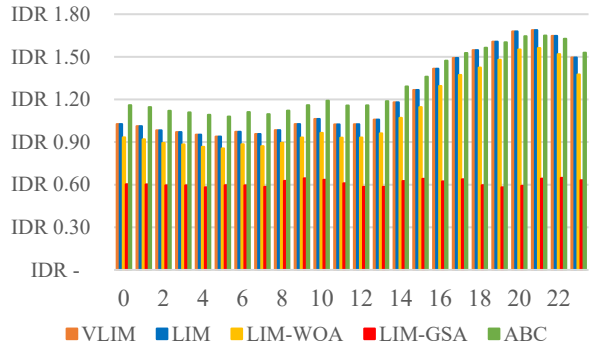

Fig. 3: Test System 2 Result in Term Cost

Table 7. Cost at 4,280.712 MW

Method	Cost
VLIM	1,026,864,483.34
LIM	1,026,864,483.34
LIM-WOA	933,084,801.10
LIM-GSA	603,026,008.08
ABC	1,159,737,275.95

Fig. 3 shows the relationship between total cost (in Billions IDR-Indonesian Rupiah) and demand at different hours. LIM-GSA achieves the lowest total cost but fails to meet the power balance constraint. LIM-WOA follows with a relatively low total cost but violates the power limits constraint. In contrast, VLIM and LIM algorithms offer competitive total costs while satisfying all the specified constraints. Despite achieving lower total costs, the failure of LIM-GSA and LIM-WOA to meet certain constraints raises concerns about system stability. Overall, VLIM and LIM algorithms provide cost-effective solutions while ensuring compliance with all constraints, promoting economic efficiency and operational reliability in the power generation system. As indicated in Table 7, which presents a cost comparison at a load of 4,280.712 MW.

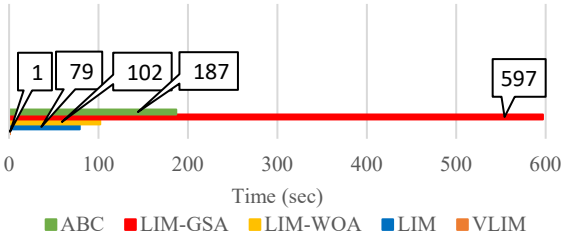

Fig. 4: Test System 2 Result in Term Computational Time

In Fig. 4, the graph represents the computational time in seconds, with the y-axis indicating the duration. It is worth noting that the result obtained shows that the VLIM algorithm demonstrates remarkable efficiency, as it completes the computation process within a mere 1 second. This finding highlights the computational advantage of VLIM, indicating its ability to quickly perform the necessary calculations and deliver results promptly. Such efficiency in computational time can contribute to enhanced productivity and time-saving in practical applications. The ability of VLIM to achieve such a swift computational time suggests its suitability for time-sensitive tasks or scenarios where real-time decision-making is crucial. By significantly reducing the computational duration, VLIM offers a practical solution that enables efficient and expedited processes within the power generation system.

Table 8. Test System 2 Result in Term Violated Constraints

Method	VLIM	LIM	LIM-WOA	LIM-GSA	ABC
PB	✓	✓	✓	X	✓
GL	✓	✓	X	✓	X
RR	✓	✓	✓	✓	✓

Table 8 provides valuable insights into the constraints violated by the evaluated lambda iteration variants. However, the results reveal an intriguing finding: both lambda iteration variants successfully fulfill all the

predetermined constraints. This highlights the effectiveness and reliability of the lambda iteration approach in maintaining a stable and efficient power generation system. The ability of these variants to satisfy the constraints showcases their robustness and suitability for practical implementation. These findings underscore the lambda iteration variants as reliable and viable solutions for optimizing power system operations while ensuring compliance with the necessary constraints.

In summary, the fact that both lambda iteration variants satisfy all the predetermined constraints strengthens their position as powerful tools for addressing the complexities of power system optimization. Their ability to maintain compliance with the constraints showcases their practicality and reliability in real-world applications. By effectively balancing the system requirements, these lambda iteration variants contribute to the stability, efficiency, and overall performance of power generation systems.

4.3. Test System 3

Figure5 displays the correlation between power production (in Giga-watts) and the corresponding demand at various hours. The x-axis indicates the range of power production values, while the y-axis represents the specific hour of the demand. Upon analyzing the findings portrayed in the graph, it becomes apparent that, apart from LIM-GSA, the algorithms did not achieve a perfect 100% accuracy in meeting the power balance constraint. Consequently, there were instances where the power generated did not precisely match the demand requirements for certain hours. As depicted in Table 9, which delineates the scheduling of power generators during the initial hour load, the recorded amount stands at 8,761.424 MW.

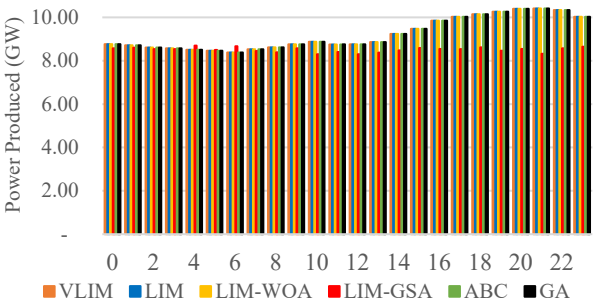

Fig. 5: Test System 3 Result in Term Power Production

Table 9. Power Allocation at 8,761.424 MW

Unit	VLIM	LIM-WOA	LIM-GSA	GA	ABC
1	250	250	285.8519	270.6148	258.4958
2	250	250	262.2004	250.3665	236.8897
3	250	250	250	260.3772	267.6668
4	250	250	250	252.1092	243.1704
5	408	408	423.1559	413.3309	448.8197
6	408	408	513.4002	453.3078	479.1705
7	408	408	485.5297	444.8807	446.8723
8	174.6	174.6	87.5	107.4275	100.5492
9	174.6	174.6	87.5	126.7554	118.6464
10	174.6	174.6	133.9282	94.89961	139.9744
11	174.6	174.6	87.5	157.6882	94.2436
12	80	114.6	64.81972	68.49597	37.96943
13	80	80	40	55.96461	38.39588
14	157.012	128.462	75	81.66251	103.5743
15	157.012	128.462	75	82.74894	122.7099
16	220	220	231.3152	286.4929	331.8211
17	220	220	220	233.6768	286.9352
18	220	220	254.3156	255.2989	347.7212
19	16	100	5	6.184078	10.16436
20	16	16	5	11.39076	10.11077
21	118.5	118.5	60	91.08253	60.98361
22	118.5	118.5	60	66.32505	66.45651
23	118.5	118.5	60	63.65653	68.03774
24	118.5	118.5	60	67.14255	91.06848
25	118.5	118.5	60	62.93813	89.42571
26	118.5	118.5	60	69.37002	63.48134
27	118.5	118.5	60	108.7538	53.67243
28	118.5	118.5	60	62.42682	89.06186
29	400	298.5	529.0811	400.6116	430.3674
30	400	400	400	489.8889	388.2568
31	400	400	400	436.143	365.4834
32	400	400	515.6651	411.5295	342.3962
33	120	160	120	135.3162	109.129
34	200	200	281.3627	242.8245	244.4801
35	200	200	263.838	305.8262	322.9066
36	345	345	345	373.4229	537.4477
37	180	180	180	207.6928	169.3011
38	200	200	331.5388	274.2359	215.1671
39	200	200	200	204.877	232.7719
40	180	180	180	180.601	166.2487
41	250	250	257.2399	286.6258	283.2957
42	250	250	250	306.4751	248.0836
Total	8761.424	8761.424	8570.742	8761.439	8761.424

Nonetheless, it is important to note that despite the lack of complete accuracy, several algorithms exhibited a significant degree of success in satisfying the power balance constraint. Notably, the VLIM, LIM, LIM-WOA, ABC, and GA algorithms demonstrated their ability to fulfill the power balance requirement within an acceptable range. Although these algorithms did not attain absolute precision, they effectively ensured that the power production was closely aligned with the demand, thereby minimizing notable deviations or imbalances. Consequently, they proved to be effective in maintaining a relatively stable power generation system where the production closely corresponds to the prevailing demand.

In Fig. 6, the graph illustrates the correlation between the total cost (in IDR-Indonesian Rupiah) and the demand at different hours. Notably, the LIM-WOA algorithm stands out by achieving the lowest total cost among the evaluated algorithms. However, it is important to note that

LIM-WOA violates one of the specified constraints, suggesting a deviation from the optimal solution. As indicated in Table 10, which presents a cost comparison at a load of 8,761.424 MW.

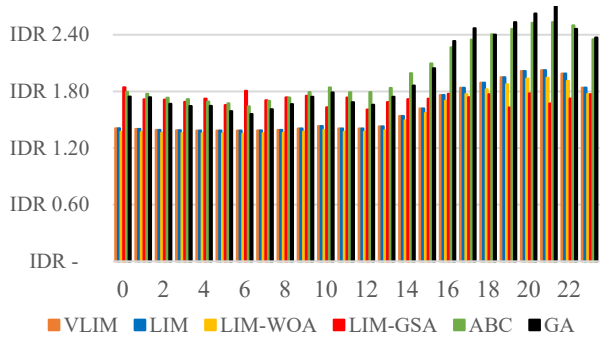

Fig. 6: Test System 3 Result in Term Cost

Table 10. Cost at 8,761.424 MW

Method	Cost (Rp)
VLIM	1,408,695,239.64
LIM	1,408,695,239.64
LIM-WOA	1,373,466,150.47
LIM-GSA	1,843,073,779.59
ABC	1,744,835,366.00
GA	1,794,132,085.85

In contrast, the two lambda iteration variants emerge as the second lowest in terms of the total cost while maintaining compliance with all the given constraints. This showcases their capability to ensure a reliable and balanced power generation system. The lambda iteration variants effectively strike a balance between cost optimization and constraint fulfillment. While LIM-WOA excels in cost reduction, its failure to adhere to a constraint raises concerns about the overall stability of the system. On the other hand, the lambda iteration variants offer competitive total costs while satisfying all the specified constraints. This highlights their suitability for real-world implementation, as they provide an optimized and dependable solution for managing power generation operations.

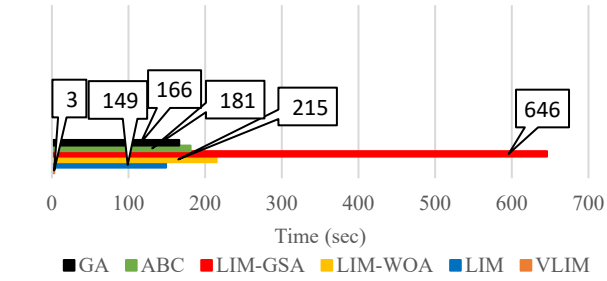

Fig. 7: Test System 3 Result in Computational Time

In Fig. 7, the graph showcases the computational time in seconds, with the y-axis representing the duration. The obtained result reveals that the VLIM algorithm exhibits impressive efficiency, completing the computation process within a mere 3 seconds. This outcome highlights the computational advantage of VLIM, underscoring its ability to swiftly perform the necessary calculations and deliver prompt results. The efficient computational time offered by VLIM can contribute to enhanced productivity and time-saving in practical applications.

The capability of VLIM to achieve such a rapid computational time indicates its suitability for time-sensitive tasks or scenarios where real-time decision-making is crucial. By significantly reducing the duration of computations, VLIM provides a practical solution that enables efficient and expedited processes within the power generation system. Overall, the findings from Fig. 7 emphasize the computational efficiency of VLIM, highlighting its potential to streamline operations and improve the overall performance of power system optimization tasks.

The time required for different computational tasks varies based on the number of units involved. For example, when we measure the time, it takes for 80 units, it amounts to 135.26 seconds. Extrapolating this to a 42-unit VLIM, we estimate that a 42-unit Dual-Population Adaptive Differential Evolution would take 67.63 seconds[29]. Similarly, if we assume a compulsion time of 9.875 seconds for 40 units, a 42-unit Squirrel Search Algorithm would take longer than 9.875 seconds [30]. Likewise, if 30 units require 130.5 seconds, then a 42-unit Adaptive Differential Evolution-Simulated Annealing would exceed 130.5 seconds [31]. When we estimate that 40 units need 233 seconds, we can infer that a 42-unit Ameliorated Grey Wolf Optimization would require more than 233 seconds[32]. Assuming a computation time of 8.6 seconds for a 40-unit system, solving a 42-unit problem with the Robust Learning Grey Wolf Optimization would take more than 8.6 seconds[33]. If we consider 40 units taking 32.5984 seconds, then a 42-unit Surrogate-Assisted Adaptive Bat Algorithm would exceed 32.5984 seconds[34]. Finally, if 40 units necessitate 5.0169 seconds, a 42-unit Improved Bat Algorithm would require more than 5.0169 seconds[35]. If we determine that a 40-unit VLIM necessitates 20 seconds for computation under the compulsion time, it can be inferred that a 42-unit Hybrid Artificial Algae Algorithm would require more than 20 seconds for the same task[36]. Similarly, if we establish that 84 units demand 76.65 seconds for computation, extrapolating to a 42-unit VLIM suggests that a 42-unit Multi-Player Harmony Search would take approximately 38.325 seconds[37]. Furthermore, when we take into account that 40 units can be computed in 7.0656 seconds, it can be surmised that a 42-unit utilizing the Differential Evolution and Gain-Sharing Knowledge-based algorithm would also require more than 7.0656 seconds for computation[38]. This shows us how fast VLIM is among all those algorithms for the same number of generators.

In the fast-paced energy market, where prices fluctuate rapidly, the ability to quickly calculate optimal solutions

for the economic dispatch problem is invaluable. VLIM's faster computation times allow market participants to respond swiftly to changing conditions, enabling them to make more informed and profitable bidding decisions. Moreover, the efficient and accurate energy bidding facilitated by VLIM leads to better resource allocation. By minimizing production costs and maximizing output, power generation systems can operate more efficiently, reducing wastage and promoting sustainability.

Table 11. Test System 3 Result in Term Violated Constraints

Constraint	VLIM	LIM	LIM-WOA	LIM-GSA	ABC
PB	✓	✓	✓	X	✓
GL	✓	✓	X	✓	X
RR	✓	✓	✓	✓	✓

Table 11 provides insightful information about the constraints violated by the lambda iteration variants that were evaluated. However, the results unveil an intriguing discovery: both lambda iteration variants successfully meet all the predetermined constraints. This emphasizes the effectiveness and reliability of the lambda iteration approach in maintaining a stable and efficient power generation system. The ability of these variants to satisfy the constraints highlights their robustness and suitability for practical implementation. These findings underscore the lambda iteration variants as reliable and practical solutions for optimizing power system operations while ensuring adherence to necessary constraints.

Fulfilling all the predetermined constraints reinforces their significance as powerful tools for tackling the intricacies of power system optimization. Their capacity to uphold compliance with the constraints demonstrates their practicality and dependability in real-world applications. By effectively balancing system requirements, these lambda iteration variants contribute to the stability, efficiency, and overall performance of power generation systems.

5. Conclusion

The paper introduces a proposed idea to improve the convergence speed of the lambda iteration method. This technique is applied to solve the economic dispatch problem in a power generating system that consists of 15 units, 16 units, and 42 units, without considering transmission losses. The study shows that the proposed VLIM (Vectorized Lambda Iteration Method) not only yields the lowest cost but also exhibits significantly faster computation time while adhering to relevant constraints. In test system 1, when compared to the fastest benchmark algorithm, VLIM reduced the computation time from 0.1 seconds to 0.004 seconds, resulting in a time reduction of up to 96%. In test system 2, VLIM reduced the computation time from 79 seconds to 1 second, achieving a time reduction of up to 96%. Similarly, in test system 3, VLIM reduced the computation time from 149 seconds to 3 seconds, attaining a time reduction of up to 98%.

The promising implications of the research's outcomes are particularly significant in the context of energy bidding, especially during hourly energy bidding processes where time plays a crucial role. In the energy industry, efficient and timely decision-making is vital to optimizing resource allocation and cost-effectiveness in power generation systems. With the adoption of the Vectorized Lambda Iteration Method (VLIM), the study demonstrates remarkable reductions in computation time compared to existing benchmark algorithms. The substantial time reductions of up to 96% and even 98% in some cases empower energy market participants to perform real-time bidding and scheduling with greater speed and precision.

Furthermore, the research's findings hold promise for addressing larger and more complex power systems with even greater benefits in terms of computation time and cost optimization. As advancements in computing technologies continue, the application of efficient algorithms like VLIM will play a vital role in shaping the future of the energy sector, leading to a more sustainable and cost-efficient energy landscape.

Acknowledgements

This research has received support from the Directorate of Research and Community Service and the Graduate Program at Telkom University. We extend our gratitude to PLN for providing the dataset used in this study, sourced from research funded by PT. PLN Persero under Memorandum of Understanding No. 006.Pj/HKM.02.01/C30000000/2022 and 034/SAM4/RC DBFJ2022, along with the Letter of Assignment from PT PLN Persero No. 1 1 5 2/L IT.00.02/C30000000/2022.

References

1) E. Obio, S. Ali, I. Oyebanjo, D. Abara, F. Suleiman, P. Ohiero, D. Oku, and V. Ogar, “Comparison of economic dispatch, opf and security constrained-opf in power system studies r<sup>n<sup>,” *J. Power Energy Eng.*, 10 (*08*) 54–74 (2022). doi:10.4236/jpee.2022.108005.
2) M. Prabavathi, and R. Gnanadass, “Energy bidding strategies for restructured electricity market,” *Int. J. Electr. Power Energy Syst.*, 64 956–966 (2015). doi:10.1016/j.ijepes.2014.08.018.
3) N.M. Azkiya, A.G. Abdullah, and H. Hasbullah, “Economic dispatch and operating cost optimization for thermal power in 500 kv system using genetic algorithm (ga),” *IOP Conf. Ser. Mater. Sci. Eng.*, 434 (*1*) (2018). doi:10.1088/1757-899X/434/1/012013.
4) M.B. Nappu, A. Arief, S.W. Soalehe, and M. Rianty, “Economic dispatch of jeneponto thermal power plant for primary energy efficiency,” *J. Phys. Conf.*

Ser., 1341 (*5*) (2019). doi:10.1088/1742-6596/1341/5/052017.

5) J. Raharjo, H. Zein, and K.B. Adam, "Optimal economic load dispatch with prohibited operating zones using large to small area technique," *Int. J. Energy Convers.*, 9 (*1*) 29–34 (2021). doi:10.15866/IRECON.V9I1.19548.
6) H. Zein, J. Raharjo, and I.R. Mardiyanto, "A method for completing economic load dispatch using the technique of narrowing down area," *IEEE Access*, 10 30822–30831 (2022). doi:10.1109/ACCESS.2022.3158928.
7) I.A. Aditya, A.A. Simaremare, J. Raharjo, Suyanto, and I. Wijayanto, "Komodo mlipir algorithm to solve generator scheduling problems," *Proc. - 2022 2nd Int. Conf. Electron. Electr. Eng. Intell. Syst. ICE3IS 2022*, 84–88 (2022). doi:10.1109/ICE3IS56585.2022.10010294.
8) R.A. Abttan, A.H. Tawafan, and S.J. Ismael, "Economic dispatch by optimization techniques," *Int. J. Electr. Comput. Eng.*, 12 (*3*) 2228–2241 (2022). doi:10.11591/ijece.v12i3.pp2228-2241.
9) G. Xiong, J. Zhang, X. Yuan, D. Shi, Y. He, Y. Yao, and G. Chen, "A novel method for economic dispatch with across neighborhood search: a case study in a provincial power grid, china," *Complexity*, 2018 (2018). doi:10.1155/2018/2591341.
10) D. Santra, A. Mukherjee, K. Sarker, and S. Mondal, "Dynamic economic dispatch using hybrid metaheuristics," *J. Electr. Syst. Inf. Technol.*, 7 (*1*) (2020). doi:10.1186/s43067-020-0011-2.
11) D.B. Miracle, R.K. Viral, P.M. Tiwari, and M. Bansal, "Hybrid metaheuristic model for optimal economic load dispatch in renewable hybrid energy system," *Int. Trans. Electr. Energy Syst.*, 2023 10–12 (2023). doi:10.1155/2023/5395658.
12) Z. Hu, C. Dai, and Q. Su, "Adaptive backtracking search optimization algorithm with a dual-learning strategy for dynamic economic dispatch with valve-point effects," *Energy*, 248 123558 (2022). doi:10.1016/J.ENERGY.2022.123558.
13) M.U. Malik, M.M. Zaid, and A. Ahmad, "Solution of economic load dispatch problem by modified whales optimization algorithm," *RAEE 2019 - Int. Symp. Recent Adv. Electr. Eng.*, (*August*) (2019). doi:10.1109/RAEE.2019.8887044.
14) K. Rehman, and A. Ahmed, "A novel hybrid moth flame optimization with sequential quadratic programming algorithm for solving economic load dispatch problem," *Mehran Univ. Res. J. Eng. Technol.*, 38 (*1*) 129–142 (2019). doi:10.22581/muet1982.1901.11.
15) S. Basak, B. Bhattacharyya, and B. Dey, "Dynamic economic dispatch using hybrid csajaya algorithm considering ramp rates and diverse wind profiles," *Intell. Syst. with Appl.*, 16 (*October 2021*) 200116 (2022). doi:10.1016/j.iswa.2022.200116.
16) S. Kumar, P.K. Singhal, V. Kumar, and L.K. Sagar, "Optimization of power flow using ga fuzzy approach," *Evergr. Jt. J. Nov. Carbon Resour. Sci. Green Asia Strateg.*, 10 (*03*) 1549–1557 (2023). doi.org/10.5109/7151702
17) M.A. Berawi, S.A.O. Siahaan, Gunawan, P. Miraj, and P. Leviakangas, "Determining the prioritized victim of earthquake disaster using fuzzy logic and decision tree approach," *Evergr. Jt. J. Nov. Carbon Resour. Sci. Green Asia Strateg.*, 7 (*2*) 246–252 (2020). doi:10.5109/4055227.
18) S. Pant, and Gaurav Bhandari, "Multi objective scheduling of chp based microgrid using," *Evergr. Jt. J. Nov. Carbon Resour. Sci. Green Asia Strateg.*, 10 (*03*) 1422–1429 (2023). doi.org/10.5109/7151691
19) A. Aurasopon, and C. Takeang, "Hybrid of lambda iteration and meta-heuristic methods for solving economic dispatch problem," *Prz. Elektrotechniczny*, 96 (*6*) 26–32 (2020). doi:10.15199/48.2020.06.05.
20) A.P. Hemmasian, K. Meidani, S. Mirjalili, and A. Barati Farimani, "VecMetaPy: a vectorized framework for metaheuristic optimization in python," *Adv. Eng. Softw.*, 166 103092 (2022). doi:10.1016/j.advengsoft.2022.103092.
21) S. Guelton, J. Falcou, and P. Brunet, "Exploring the vectorization of python constructs using pythran and boost simd," *WPMVP 2014 - Proc. 2014 ACM SIGPLAN Work. Program. Model. SIMD/Vector Process. Co-Located with PPoPP 2014*, 79–86 (2014). doi:10.1145/2568058.2568060.
22) N. Watkinson, P. Tai, A. Nicolau, and A. Veidenbaum, "NumbaSummarizer: a python library for simplified vectorization reports," *2020 IEEE 34th Int. Parallel Distrib. Process. Symp. Work. IPDPSW 2020*, 269–275 (2020). doi:10.1109/IPDPSW50202.2020.00058.
23) J. Raharjo, A. Soeprijanto, and H. Zein, "Multi dimension of coarse to fine search method development for solving economic dispatch," *Indones. J. Electr. Eng. Comput. Sci.*, 3 (*1*) 1–9 (2016). doi:10.11591/ijeecs.v3.i1.pp1-9.
24) K.M.D. Puspitasari, J. Raharjo, A.S. Sastrosubroto, and B. Rahmat, "Generator scheduling optimization involving emission to determine emission reduction costs," *Int. J. Eng. Trans. B Appl.*, 35 (*8*) 1468–1478 (2022). doi:10.5829/IJE.2022.35.08B.02.
25) J.L.K. Grace, A. Maneengam, P.K.V. Kumar, and J. Alanya-Beltran, "Design and implementation of machine learning modelling through adaptive hybrid swarm optimization techniques for machine management," *Evergr. Jt. J. Nov. Carbon Resour. Sci. Green Asia Strateg.*, 10 (*2*) 1120–1126 (2023). doi:10.5109/6793672.
26) M.M. Rahman, S. Saha, M.Z.H. Majumder, T.T. Suki, M.H. Rahman, F. Akter, M.A.S. Haque, and M.K. Hossain, "Energy conservation of smart grid system using voltage reduction technique and its challenges," *Evergr. Jt. J. Nov. Carbon Resour. Sci. Green Asia*

Strateg., 9 (*4*) 924–938 (2022). doi:10.5109/6622879.
27) G. Chauhan, A. Jain, and N. Verma, "Solving economic dispatch problem using mipower by lambda iteration method," *2017 1st Int. Conf. Intell. Syst. Inf. Manag.*, 95–99 (2017). doi:10.1109/ICISIM.2017.8122155.
28) P.K. Singhal, R. Naresh, V. Sharma, and N. Goutham Kumar, "Enhanced lambda iteration algorithm for the solution of large scale economic dispatch problem," *Int. Conf. Recent Adv. Innov. Eng. ICRAIE 2014*, 0–5 (2014). doi:10.1109/ICRAIE.2014.6909294.
29) X. Chen, "Novel dual-population adaptive differential evolution algorithm for large-scale multi-fuel economic dispatch with valve-point effects," *Energy*, 203 117874 (2020). doi:10.1016/j.energy.2020.117874.
30) S. V.P, S. M, and S. P.D, "Large-scale economic load dispatch using squirrel search algorithm," *Int. J. Energy Sect. Manag.*, 14 (*6*) 1351–1380 (2020). doi:10.1108/IJESM-02-2020-0012.
31) D. He, L. Yang, and Z. Wang, "Adaptive differential evolution based on simulated annealing for large-scale dynamic economic dispatch with valve-point effects," *Math. Probl. Eng.*, 2018 (2018). doi:https://doi.org/10.1155/2018/4745192.
32) D. Singh, and J.S. Dhillon, "Ameliorated grey wolf optimization for economic load dispatch problem," *Energy*, 169 398–419 (2019). doi:10.1016/j.energy.2018.11.034.
33) T. Tai, C. Lee, and C. Kuo, "A hybrid grey wolf optimization algorithm using robust learning mechanism for large scale economic load dispatch with vale-point effect," *Appl. Sci.*, 13 (*4*) (2023). doi:10.3390/app13042727.
34) A. Pang, H. Liang, C. Lin, and L. Yao, "A surrogate-assisted adaptive bat algorithm for large-scale economic dispatch," *Energies*, 16 (*2*) (2023). doi:10.3390/en16021011.
35) W. Yang, R. Li, Y. Yuan, and X. Mou, "Economic dispatch using modified bat algorithm," *Front. Energy Res.*, 10 (*September*) 1–15 (2022). doi:10.3389/fenrg.2022.977883.
36) M. Kumar, and J.S. Dhillon, "Hybrid artificial algae algorithm for economic load dispatch," *Appl. Soft Comput. J.*, 71 89–109 (2018). doi:10.1016/j.asoc.2018.06.035.
37) M. Nazari-Heris, B. Mohammadi-Ivatloo, S. Asadi, and Z.W. Geem, "Large-scale combined heat and power economic dispatch using a novel multi-player harmony search method," *Appl. Therm. Eng.*, 154 (*March*) 493–504 (2019). doi:10.1016/j.applthermaleng.2019.03.095.
38) Q. Liu, G. Xiong, X. Fu, A.W. Mohamed, J. Zhang, M.A. Al-Betar, H. Chen, J. Chen, and S. Xu, "Hybridizing gaining–sharing knowledge and differential evolution for large-scale power system economic dispatch problems," *J. Comput. Des. Eng.*, 10 (*2*) 615–631 (2023). doi:10.1093/jcde/qwad008.