energies

MDPI

Article

Fully Decentralized, Cost-Effective Energy Demand Response Management System with a Smart Contracts-Based Optimal Power Flow Solution for Smart Grids

Yaçine Merrad [1], Mohamed Hadi Habaebi [1], Siti Fauziah Toha [2,*], Md. Rafiqul Islam [1], Teddy Surya Gunawan [3] and Mokhtaria Mesri [4]

[1] IoT & Wireless Communication Protocols Laboratory, Department of Electrical & Computer Engineering, Kulliyyah of Engineering (KOE), International Islamic University Malaysia, Jalan Gombak, Kuala Lumpur 53100, Malaysia; yacinechoupot@yahoo.fr (Y.M.); habaebi@iium.edu.my (M.H.H.); rafiq@iium.edu.my (M.R.I.)
[2] Department of Mechatronics, Kulliyyah of Engineering (KOE), International Islamic University Malaysia, Jalan Gombak, Kuala Lumpur 53100, Malaysia
[3] Department of Electrical & Computer Engineering, Kulliyyah of Engineering (KOE), International Islamic University Malaysia, Jalan Gombak, Kuala Lumpur 53100, Malaysia; tsgunawan@iium.edu.my
[4] Department of Electronics, University Amar Télidji of Laghouat, Laghouat 03000, Algeria; m.mesri@lagh-univ.dz
* Correspondence: tsfauziah@iium.edu.my

Abstract: Recent advances in control, communication, and management systems, as well as the widespread use of renewable energy sources in homes, have led to the evolution of traditional power grids into smart grids, where passive consumers have become so-called prosumers that feed energy into the grid. On the other hand, the integration of blockchain into the smart grid has enabled the emergence of decentralized peer-to-peer (P2P) energy trading, where prosumers trade their energy as tokenized assets. Even though this new paradigm benefits both distribution grid operators and end users in many ways. Nevertheless, there is a conflict of interest between the two parties, as on the one hand, prosumers want to maximize their profit, while on the other hand, distribution system operators (DSOs) seek an optimal power flow (OPF) operating point. Due to the complexity of formulating and solving OPF problems in the presence of renewable energy sources, researchers have focused on mathematical modeling and effective solution algorithms for such optimization problems. However, the control of power generation according to a defined OPF solution is still based on centralized control and management units owned by the DSO. In this paper, we propose a novel, fully decentralized architecture for an OPF-based demand response management system that uses smart contracts to force generators to comply without the need for a central authority or hardware.

Keywords: decentralized; blockchain; optimal power flow; smart grid; smart contracts

Citation: Merrad, Y.; Habaebi, M.H.; Toha, S.F.; Islam, M.R.; Gunawan, T.S.; Mesri, M. Fully Decentralized, Cost-Effective Energy Demand Response Management System with a Smart Contracts-Based Optimal Power Flow Solution for Smart Grids. *Energies* **2022**, *15*, 4461. https://doi.org/10.3390/en15124461

Academic Editor: Abu-Siada Ahmed

Received: 23 May 2022
Accepted: 17 June 2022
Published: 19 June 2022

1. Introduction

The current tremendous expansion of distributed energy sources and the democratization of electricity generation through conventional and renewable sources such as wind turbines, solar energy, and advances in energy storage capacity [1,2], combined with the advancement of digital communication and control technologies, have enabled the transition from traditional grid systems to what is known as a smart grid, in which traditional consumers have evolved into proactive prosumers that can join a grid branch of the larger distribution network and contribute to energy demand management within the grid they have joined, either with traditional fuel-based energy generators or renewable sources [3,4]. In addition, the integration of blockchain into the smart grid and the functionalities of smart contracts enabled the emergence of peer-to-peer energy trading [5,6], where prosumers can trade their energy generated and injected into the grid in a decentralized manner. It should

be noted that the reduction of influence and control by a central authority is paramount in all blockchain-based applications [7]. Similarly, the decentralization of Blockchain-based P2P energy trading platforms and the safe reduction of the authority of the DSO is an aimed target [8,9]. Since prosumers mainly use renewable energy sources, energy and climate policies promote and support such energy platforms to meet both increasing energy demand and sustainable climate goals [10,11]. On the other hand, end users can reduce their electricity costs and gain lucrative benefits, making such platforms a win-win scenario. In P2P energy trading platforms, prosumers are rewarded accordingly for their delivered energy through crypto-tokens, which are either fungible or non-fungible [12] and delivered through appropriately designed smart contracts. In an integrated system, the utility has complete control over the location and connection of distributed energy resources (DER) and will seek to optimally manage demand, while in unbundled systems, each prosumer seeks its own benefit and its goal is to maximize its lucrative profit, according to market rules. Thus, from this perspective, a prosumer that joins a grid and the grid operator can potentially have conflicting goals. The DSO analyzes grid operations and grid investments in terms of peak power flows, while the prosumer sees its revenue in terms of aggregated energy exchange. Moreover, the flexibility of the smart grid, where new prosumers can constantly join or leave the grid, poses a challenge to the effective implementation of demand response control and management systems, which require less flexible deployment of hardware [13]. Moreover, a demand response control and management system deployed by DSOs would not be considered a decentralized solution and would run counter to the consortium-based decision-making principle of blockchain applications. One of the most widely used demand response solutions is the OPF-based control system. The OPF solution aims to achieve a steady-state operating point in the grid that reduces power generation costs while satisfying demand and operating constraints [14]. The approach to optimize the power injection within grids considering thresholds is generally challenging because it is a nonlinear and non-convex optimization problem. With renewable energy sources in the grid, the OPF problem becomes even more complicated in both formulation and solution because the generation capacities of the renewable energy sources, which are part of the constraints of the problem, are unpredictable [15]. Therefore, many researchers have addressed the formulation and solution of the OPF problem for hybrid grids and presented a number of ingenious models [16,17]. Most of the works have focused on presenting realistic mathematical models or algorithms to solve the OPF problem for hybrid power sources. However, beyond that, little work has been conducted on how the computed OPF solution can be used for decentralized power generation control in smart grids. So far, the established scheme is to use control units (hardware and software), which does not go in the direction that aims to minimize DSO authority. In this paper, a novel decentralized, transparent, and secure OPF model is used to locally coordinate power generation in distributed networks while taking into account network constraints, without the need for a centralized control unit. The paper describes a detailed approach for implementing the decentralized OPF on a private blockchain smart contracts platform that enables an immutable and access-controlled transaction system for tokenized power assets. The model solves the OPF problem of a given grid where all constraints and fixed parameters are set within an immutable and autonomous smart contract. The only variable parameter is the load demand, which can be updated in the smart contract by a load monitoring unit, either periodically [18] or in real time [19] or even using short-term load forecasting [5,20], which would allow the smart contract to operate without any required outside interaction. The smart contract solves the OPF problem for a local network. The OPF solution would be computed by a decentralized, unbiased smart entity and would thus be unchallengeable. The solution would be stored in the blockchain public ledger, making it safe from tampering. Prosumers would only receive the energy tokens they are entitled to if they comply with the OPF solution of the smart contract.

The structure of the article can be outlined as follows: Section 2 highlights the motivation of the presented work, Section 3 investigates the feasibility of an on-chain solution

to the OPF problem through a case evaluation of the execution cost of an on-chain solution for an OPF model for a three-bus network using the linear approximation DC-OPF. This is to show why such an approach is not realistic. Section 4 describes the proposed new improved smart contract-based model that can be generalized for any defined OPF problem with effective execution cost. In Section 5, we compare, discuss, and comment on the measured execution costs of the models from Sections 3 and 4. Finally, we draw conclusions in Section 6.

2. Motivation

Blockchain integration in smart grids primarily aims to enable a decentralized P2P energy trading platform without central authority [21], as is the case with blockchain cyptocurrencies. This is enabled by smart contracts that enable trustworthy transactions and agreements between different parties without the need for a central authority, legal system, or external enforcement mechanism [22]. A smart contract is a self-executing code that cannot be tampered with or modified during its execution, thus obligating the interacting parties to abide by the terms agreed upon in the smart contract. However, smart contracts are not suitable for complex computational tasks because the fees for executing transactions are high and proportional to the computational resources required for execution. Therefore, in applications based on smart contracts, the computations on-chain must be minimized [23]. A study of the code complexity of smart contracts conducted in [24] found that out of 53,757 contracts analyzed, only a very small fraction of the verified smart contracts on the Ethereum blockchain (6.9%) actually fall into the Turing Complete Functions complexity class. Solving the OPF problem of a given power grid through a smart contract would force network participants to follow the computed solution in a fully decentralized manner, optimally managing energy demand without the need for a central authority. However, the OPF problem is particularly complex because it is a non-convex, nonlinear problem whose solution by numerical methods requires nondeterministic polynomial time (NP) [25]. Therefore, a chain-based solution to the OPF problem seems unrealistic. Demand response in the smart grid is mainly based on centralized systems deployed and owned by the DSO. Such solutions are not only centralized and not in line with the vision and goal of blockchain-based applications, but also prove to be ineffective. This is due to the conflicting interests of prosumers and grid operators, as discussed in Section 1, as well as the flexibility of the smart grid, where new nodes are constantly being added and leaving the network. As clearly stated in [26], energy demand response remains an open problem in current smart grids and P2P energy trading platforms, as many bus voltages exceed the voltage limit applicable to the grid. In this context, we believe that establishing a Nash game arbitrated by a decentralized smart contract to create competition among provers proposing solutions to the OPF problem, and accordingly rewarding those with the optimal proposals, would significantly reduce the on-chain computations while providing a fully decentralized energy demand management system. In such a system, the OPF is calculated off-chain, and the proposed solutions are verified and compared in the smart contract, significantly reducing the required on-chain calculations. Moreover, the proposed system is designed as a competition: The fewer provers proposed an optimal solution, the more a particular prover with an optimal solution proposal wins. This prevents both malicious cooperation between provers and lack of commitment in computing the OPF solution. This would also encourage and spur the development of more effective OPF solution algorithms, similar to how mining competition in PoW blockchains has led to a hashing race and the development of high-performance specialized mining hardware. This work aims to introduce and develop such a scheme and evaluate its feasibility by evaluating the execution cost to solve a 14-bus AC-OPF problem.

3. On-Chain-Based Solution for a 3-Bus Network, Using DC-OPF Approximation

Smart contracts are not suitable for computationally intensive problems because the execution cost of transactions is proportional to their processing complexity [27,28].

Moreover, complex code can lead to potential gaps in the source code that can be exploited by hackers to intentionally bug the system [29]. This is fatal for the blockchain application that is tailored to the smart contract in question, as flaws in smart contracts cannot be repaired since the source code cannot be modified once it is deployed. In this section, we attempt to develop and evaluate the execution cost of an on-chain OPF solution for a simple three-bus network, using the IEEE three-bus test feeder [30]. In doing so, we use a simplified approach to the OPF problem, called the DC-OPF [31] model. The goal is to evaluate how realistic it is to implement an on-chain OPF solution by comparing the execution cost for a much simpler version. Thus, if the cost for a simple version is already too high, the approach is not suitable for more complex models.

3.1. DC-OPF Model

Even though the model is called DC, it does not refer to DC high voltage generator power flow through transmission lines. Actually, the model is indeed an attempt to give a linear approximation to non-linear AC power network equations. In fact, power flow between nodes i and j in a power network is represented by Equation (1).

$$S_{i,j} = V_i^2 y_{ij}^* - V_i V_j^* y_{ij}, \tag{1}$$

where V_i and V_j are the voltage at nodes i and j, respectively, and y_{ij} is the line admittance and they are all complex quantities. The $(^*)$ operator refers to the conjugate of a complex entity. DC-OPF considers line flow constraints, as opposed to the Economic Dispatch, which is the simplest approximation to the OPF problem and which only considers the grid as only a single transmission line with no constraints on power flow between two nodes. How the transmission line is modeled in DC-OPF approximation is illustrated in Figure 1.

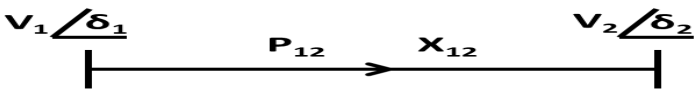

Figure 1. Transmission line model in DC-OPF.

The first step in linearizing the power flow equations is to consider only the series reactance of the line and neglect its ohmic resistances in the transmission line model π described in Section 3. Accordingly, the line active power flow between point 1 and 2 is given by Equation (2), as in Figure 1, where V_1, V_2, δ_1, δ_2 are the voltage magnitudes and angles for points 1 and 2, respectively, and X_{12} is the line reactance:

$$P_{12} = V_1 V_2 \frac{sin(\delta_1 - \delta_2)}{X_{12}}, \tag{2}$$

As one can see, Equation (2) is obviously nonlinear; to linearize it, the following assumptions are made: Voltages are nominal constants, where:

$V_1 = V_2 = 1$ Per Unit (p.u.); $sin(\delta_1 - \delta_2) = \delta_1 - \delta_2$

Thus, we obtain Equation (3):

$$P_{12} = V_1 V_2 \frac{(\delta_1 - \delta_2)}{X_{12}} = (\delta_1 - \delta_2) * b_{ij} \tag{3}$$

These assumptions are only possible for networks that are not heavily loaded. The more heavily loaded the network, the more important are the phase angles at the nodes of the network buses. The larger the angle t, the less accurate the approximation $sin(t) = t$.

Thus, the active power flow on the line is related to the voltage phase angles δ and the line susceptance b_{ij}. The power injected at a given generator bus i, connected to n

other buses via n transmission lines, and its relation to the reactive power flow of the transmission lines is described in Equation (4).

$$P_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} P_{ij} = \sum_{\substack{j=1 \\ j \neq i}}^{n} b_{ij}(\delta_i - \delta_j); i \geq 1, \tag{4}$$

The matrix notation for Equation (4) is described in Equation (5).

$$P = B\delta = \begin{bmatrix} P_1 \\ \vdots \\ P_n \end{bmatrix} = \begin{bmatrix} b_{12} + \cdots + b_{1n} & \cdots & -b_{1n} \\ \vdots & \ddots & \vdots \\ -b_{n1} & \cdots & b_{n1} + \cdots + b_{nn-1} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_n \end{bmatrix}, \tag{5}$$

B is called the bus susceptance matrix for the power network of n connected buses. B is used to calculate the DC-OPF. Note that all scalar elements of the diagonal matrix $B_{ii}, i \in [1, n]$ are calculated by the formula in Equation (6).

$$B_{ii} = \sum_{\substack{j=1 \\ j \neq i}}^{n} b_{ij} \tag{6}$$

As for the non-diagonal elements of B, where $i \neq j$, they are either null if there is no line between nodes i and j, otherwise $B_{ij} = -b_{ij}$.

The OPF objective function consists of minimizing the generators' power injection cost, subject to the voltage angles of the lines and the maximum power capacity of the generators and transmission line constraints while fulfilling the load power demand. Having defined cost functions $C(P_G)$ for all generators in the network and the equation for power flow along each transmission line that defines the relationship between power injections and voltage angles, deduced through the bus susceptance matrix, the OPF problem in a network with n generation buses can be formulated as in Equation (7):

$$F(P_G) = \sum_{i=0}^{n} C(P_{Gi}), \tag{7}$$

Equation (7) is subject to the constraints defined in Equation (8).

$$\begin{cases} B\delta = P_G - P_D, \\ H_1(P_{ij}) = \frac{1}{x_{ij}}(\delta_i - \delta_j) \leq P_{ij}^{max}, \\ H_2(P_{ij}) = \frac{1}{x_{ij}}(\delta_i - \delta_j) \geq -P_{ij}^{max}, \\ G_1(P_i) = P_{Gi} \leq P_G^{max}, \\ G_2(P_i) = P_{Gi} \geq P_G^{min}, \end{cases} \tag{8}$$

Generator cost functions define the relationship between power generated and cost. A study of how generator cost functions are derived can be found in [32]. However, the most commonly used type of generator cost function is of the quadratic form, as formulated in Equation (9):

$$C_i(P_i) = \alpha_i P_i^2 + \beta_i P_i + \gamma_i, \tag{9}$$

where $\alpha_i, \beta_i, \gamma_i$ are scalar constants. The equation of the bus susceptance matrix represents equality constraints for the OPF objective function, whereas the transmission lines and the maximum power capacity of the generators are the inequality constraints for the problem. The Lagrangian multipliers and the Hessian matrix derived from the Lagrangian function are used to solve the OPF objective function defined by the DC-OPF model for power networks. A reference bus with a fixed voltage phase angle $\delta_{ref} = 0$ must be defined for

this purpose. Equation (10) defines the Lagrangian of the DC model OPF problem in a network of N_b generation buses and M transmission lines.

$$\begin{aligned} L = &\sum_{i=1}^{Nb} C(P_{Gi}) + \lambda_i(B\delta - P_G - P_D) + \sum_{i=1}^{Nb} \mu_i G_1(P_{Gi}) - \sum_{i=1}^{Nb} \mu_i^{'} G_2(P_{Gi}) \\ &+ \sum_{k=1}^{M} \beta_k H_1(P_{Gm}) - \sum_{k=1}^{M} \beta_k^{'} H_2(P_{Gm}), \end{aligned} \tag{10}$$

Considering only the problem equality constraints, the set of linear equations derived from the Lagrangian function in Equation (10) is as defined in Equation (11).

$$\begin{cases} \dfrac{dL}{dP_{G1}} = 0; \cdots \dfrac{dL}{dP_{Gnb}} = 0, \\ \dfrac{dL}{d\lambda_1} = 0; \cdots \dfrac{dL}{d\lambda_{nb}} = 0, \\ \dfrac{dL}{d\delta_1} = 0; \cdots \dfrac{dL}{d\delta_{nb}} = 0, \end{cases} \tag{11}$$

Inequality constraints need to fullfill the Karush Khun Tuker conditions [33]. This means that for every inequality constraint defined, there are two possibilities. Either the inequality constraint is satisfied at the boundary condition and actually functions as an equality constraint, and its Lagrange multiplier is nonzero. If the inequality condition is inactive, it does not matter; its Lagrange multiplier is zero. In Equation (10), we have $(2N_b + 2M)$ inequality constraints with two possibilities each, thus, allowing $2^{(2N_b+2M)}$. Accordingly, for each combination, the active constraint is considered as an equality constraint and Equation (11) is updated accordingly. The solution corresponding to a particular combination must satisfy all system constraints and all computed Lagrange multipliers must be greater than or equal to zero. Otherwise, the respective solution would be considered invalid.

3.2. Implementation for a 3-Bus DC-OPF Problem

To control a power grid in a decentralized manner using a smart contract, a smart contract must be developed and deployed for a specific grid. First, a reference bus must be selected. To ensure decentralization and fairness, the selection for the reference bus should be cyclic. However to simplify the implementation, we fixed the reference bus to bus 1 for the network illustrated in Figure 2. In a power system, all parameters are fixed except the load, which varies. The load can be monitored in real time or periodically and updated to the controlling smart contract or can even be pre-implemented in the smart contract using forecasted data. The implementation was done using hourly updating of the load. Off-chain load monitoring unit calls the smart contract periodically to update the load. Before the load is changed, the most recently computed OPF solution is shared on the ledger biding each generator to its optimal power flow generation requirements. The OPF solution is linked to its corresponding date in *Hour*/*Day*/*Month*/*Year* format, where the hour is in the format 0 to 23, the day is in the format 0 to 6, and the month is in the format 0 to 11. The date is updated internally, on-chain in the smart contract each time the load monitoring unit updates the load demand of the network every hour by calling the *UpdateCurrentDate* function. Each time the function is called, the hour field of the smart contract's date structure is incremented. The updated date hour consists of the remainder of dividing the incremented hour value by 24, so that when 24 is reached the hour field is reset to 0 and the day field is incremented accordingly; the updated day consists of the remainder of dividing the incremented day field by 7, so that when 7 is reached the day field is reset to 0 and the month field is incremented. In the same way, the month field is reset to 0 when it reaches 12, and the year field is incremented. This serves as a non-contestable time stamp issued by an impartial party. How this is implemented in the concerned smart contract is illustrated in pseudocode in Algorithm 1.

Algorithm 1 OPF Solver Smart contract: Update DC-OPF Buses Load Demand

```
Fixed Pl1
Fixed Pl2
Fixed Pl3
Structure: Date
   Uint: Hour
   Uint: Day
   Uint: Month
   Uint: Year
End Structure
Date: CurrentDate
Address: LoadMonitoringUnit
Constructor()
    CurrentDate.Hour← hh/* set to contract deployment hour */
    CurrentDate.Day← dd/* set to contract deployment day */
    CurrentDate.Day← mm/* set to contract deployment month */
    CurrentDate.Day← yy/* set to contract deployment year */
End Constructor
Procedure UpdateCurrentDate()
   CurrentDate.Hour ← (CurrentDate.Hour + 1)% 24
   IfCurrentDate.Hour == 0 Then
      CurrentDate.Day ← (CurrentDate.Day + 1)% 7
      If CurrentDate.Day==0 Then
         CurrentDate.Month ←CurrentDate.(Month + 1)% 12
         If CurrentDate.Month==0Then
            CurrentDate.Year ← CurrentDate.Year + 1
         End If
      End If
   End If
End Procedure
Procedure UpdateBusesLoadDemand(Fixed Pl1, Fixed Pl2, Fixed Pl3)
   Pl1 ← Pl1
   Pl1 ← Pl2
   Pl1 ← Pl3
   UpdateCurrentDate()
End Procedure
```

The example aimed to be solved is a three-buses network with a load and generation line on each bus connected in a ring topology, as shown in Figure 2. The generation cost function for all generators in the network is of the quadratic form, as in Equation (9).

The generation cost functions' parameter for each generator and their generation capacity, as well as the network parameters, are defined in Tables 1 and 2, respectively.

Table 1. Generators cost coefficients and generation capacity.

Unit	Bus	Cost Coefficients			P_{gmin}	P_{gmax}
		a	b	c	(MW)	
G_3	3	118.8206	37.8896	0.01433	5.0	20.0
G_2	2	218.3350	18.1000	0.00612	10.0	150.0
G_1	1	142.7348	10.6940	0.00463	20.0	200.0

Table 2. Transmission lines parameters.

Line	Resistance	Reactance (P.u.)	Flow Limit (MW)
1–2	0.0	0.20	55.0
1–3	0.0	0.40	55.0
2–3	0.0	0.25	55.0

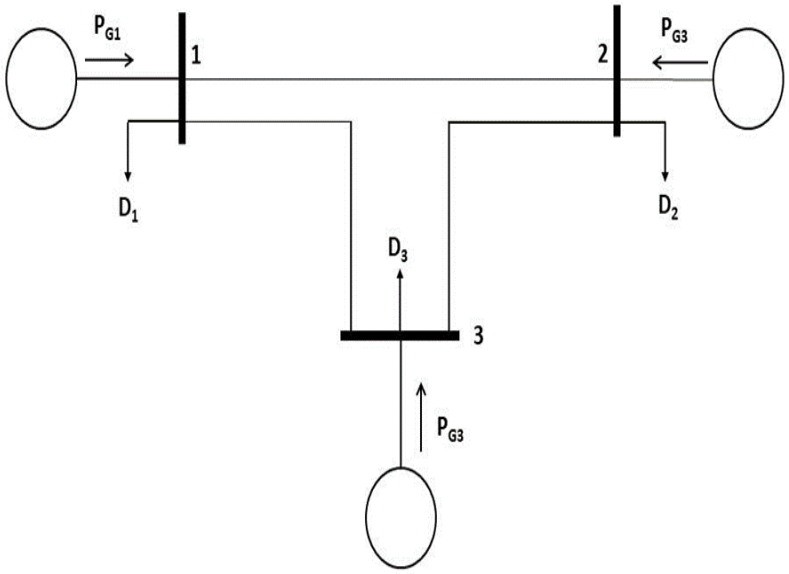

Figure 2. Three-bus power network.

The network bus susceptance matrix B is defined in Equation (12).

$$\begin{pmatrix} B_{\times 11} & B_{\times 12} & B_{\times 13} \\ B_{\times 21} & B_{\times 22} & B_{\times 23} \\ B_{\times 31} & B_{\times 32} & B_{\times 33} \end{pmatrix} = \begin{pmatrix} 1/x_{12} + 1/x_{13} & -1/x_{12} & 1/x_{13} \\ -1/x_{12} & 1/x_{12} + 1/x_{23} & -1/x_{23} \\ -1/x_{13} & -1/x_{23} & 1/x_{13} + 1/x_{23} \end{pmatrix} \tag{12}$$

The Lagrange function for the system is given in Equation (13).

$$L = \sum_{i=1}^{Nb} (a_i + b_i P_{Gi} + c_i P_{Gi}^2) + \sum_{i=1}^{Nb} \lambda_i (\sum_{j=1}^{Nb} \beta_{ij} \gamma_j - P_{Gi} + P_{ij} + \sum_{k=1}^{M} \mu_i (P_{Gi} - P_G^{max})) \tag{13}$$

From Equation (13), we deduce Equation (14), where the partial derivative of the Lagrangian function with respect to each variable is null.

$$\begin{cases} \frac{dL}{d\mu_1} = 0; \cdots \frac{dL}{d\mu_{nb}} = 0, \\ \frac{dL}{d\mu_1^{'}} = 0; \cdots \frac{dL}{d\mu_{nb}^{'}} = 0, \\ \frac{dL}{d\beta_1} = 0; \cdots \frac{dL}{d\beta_{nb}} = 0, \\ \frac{dL}{d\beta_1^{'}} = 0; \cdots \frac{dL}{d\beta_{nb}^{'}} = 0, \end{cases} \tag{14}$$

From Equation (14), we derive 24 linear equations with 24 variables. Since the load on each bus is the only variable parameter in the network, the goal is to come up with a system

of linear equations, as described in the matrix in Equation (15), where the coefficients are functions of the buses load power demand.

$$\begin{bmatrix} Coef_{00}(P_{l1}, P_{l2}, P_{l3}) & \cdots & Coef_{0\,24}(P_{l1}, P_{l2}, P_{l3}) & Cst_1 \\ \vdots & \ddots & \vdots & \vdots \\ Coef_{24\,0}(P_{l1}, P_{l2}, P_{l3}) & \cdots & Coef_{24\,24}(P_{l1}, P_{l2}, P_{l3}) & Cst_{24} \end{bmatrix}, \tag{15}$$

where P_{l1}, P_{l2}, P_{l3} are then sent to the smart contract every hour and the coefficients of the matrix are set accordingly in the smart contract. As explained earlier, each Lagrange multiplier can assume one of two states with respect to the inequality constraints, either as an equality constraint or as neglected. Since there are 12 inequality constraints, this leads to $2^{12} = 5096$ possibilities. However, the system has a unique solution. The system of linear equations in Equation (15) is solved for each possibility by looping through them until coming up with a valid solution, as described in the flowchart in Figure 3. Solutions for which a multiplier is negative or which do not satisfy the system constraints are not considered valid.

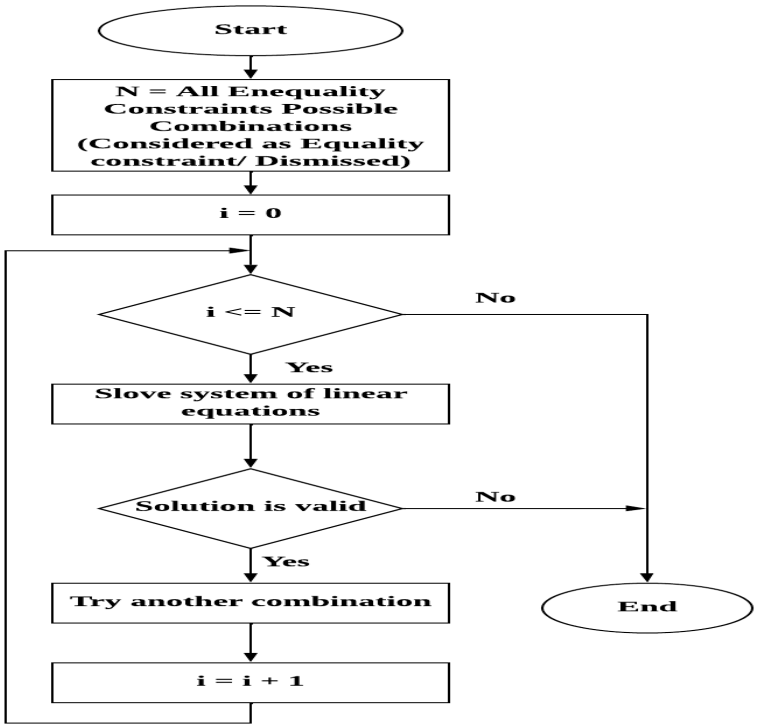

Figure 3. Solving the DC-OPF problem.

To solve each of the possible systems of linear equations, each matrix must be row-reduced so that all diagonal coefficients $Coef_{ii}$ are ones, and the rest are zeros, to obtain a matrix of the form of Equation (16).

$$\begin{bmatrix} 1 & \cdots & 0 & R_{cst_1} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & R_{cst_{24}} \end{bmatrix}, \tag{16}$$

The solution to the linear system of equations, which is the solution to our OPF objective function, is then $S = \{S_1, ...S_{23}\}$, which represents the P_G at each generator as well as the angular displacement at each end of the transmission line, as well as all the Lagrange multipliers, simply saying $S_i = R_{cst_i}$. The main function, which loops through

all the linear systems of equations corresponding to each of the possible combinations of inequality constraints, is described in the pseudocode in Algorithm 2.

Algorithm 2 OPF Solver Smart contract: Solving DC-OPF matrix (main function)

```
1:  Procedure UpdateMatrixCoeff(Fixed[][] Coeff)
2:    Uint[12] Sum /* a local array of 12 uint elements, where all indexes
    are initialized to 0 */
3:    Uint i ← 0
4:    Uint j ← 0
5:    While !(∀ s in Sum, s==1) Do
6:      While i ≤ 11 Do
7:        If Sum[i]==0
8:          DeleteRow(i + 12, Coeff) /* Delete row corresponding to
     the inequality constraint multiplier that is not considered */
9:        End If
10:       While j ≤ 12 Do
    /* Set to zero columns corresponding to the
    inequality constraint multiplier that are not considered */
11:         Coeff[i][j + 11] ← Coeff[i][j + 11] * Sum[i]
12:         j ← j + 1
13:       End While
14:       i ← i + 1
15:     End While
16:     Increment(Sum)
17:     Bool Solved ← Solve(Coeff)
18:     If Solved Then
19:       Break /* Break from the loop, system already solved */
20:     End If
21:   End While
22: End Procedure
```

The function internally calls auxiliary functions that are used to update the matrix coefficient of the linear system of equations that corresponds to the chosen combination of inequality constraints. These are described in the pseudocode in Algorithm 3. The Lagrange multipliers for inequality constraints correspond to the elements in the columns from 12 to 24 in the DC-OPF matrix defined in the smart contract and the rows from 12 to 24 corresponding to the partial Lagrange derivative with respect to the multipliers for inequality constraints.

Algorithm 3 OPF Solver Smart contract: Update DC-OPF matrix coefficients (helper functions)

```
Fixed[][]: OPFmatrixCoeff
/* The following helper function used to increment a twelve bits binary number,
representing all the inequality constraints possibilities, either
considered as equality constraint or not considered */
Procedure Increment(Uint sum[])
    Uint j ← 1
    Uint carry ← 0
    sum[0] ← (sum[0] + 1)% 2
    carry ← (sum[0] + 1)/2
    While carry !=0 ||j==11 Do
        sum[j] ← (sum[j] + carry)% 2
        carry ←(sum[j] + carry)/2
        j ← j + 1
    End While
End Procedure
/* The following helper function used to delete a row from a matrix
given its index, it serves to delete rows corresponding to inequality
constraints multipliers that are not considered */
Procedure DeleteRow(Uint index, Fixed[][] Matrix)
    Uint j ← 0
    Uint j ← 0
    While i ≤ Matrix.length() Do
        i ← i + 1
        While j ≤ Matrix[0].length() Do
            Matrix[i][j] ← Matrix[i + 1][j]
            j ← j + 1
        End While
        Matrix.pop()
    End While
End Procedure
```

The *solve* function, called at each iteration of the loop in the main function, solves the system of linear equations passed to it as a matrix, using an auxiliary function that row-reduces the passed matrix. This function is described in Algorithm 4, while the *solve* function is described in Algorithm 5. After the system is solved, an updated solution corresponding to a given total load power demand is stored in the OPF smart contract and is thus available in the public ledger of the blockchain. The solution is linked in a structure to the corresponding date of the load. When a smart meter associated with a particular generator on the grid requests an energy token from the smart contract assigned to reward prosumers, the smart contract must first verify that the generation matches the particular OPF solution by invoking the OPF Solver smart contract. If the generation does not match the most recent OPF solution, the corresponding token reward will not be granted. In such a situation, the respective prosumer or generator owner would see their energy contribution wasted. Prosumers and microgrid owners are thus forced to comply with the OPF solution without the need for a central authority or deployed hardware. This is illustrated in Figure 4.

Algorithm 4 OPF Solver Smart contract: Solve a system of linear equations using row-reduction algorithm

```
1:  Procedure (Fixed[][] Coeff)
2:    Uint N ← B.length
3:    Uint i ← 0
4:    Uint j ← 0
5:    Fixed[N][N-1] A /* Variables coefficient matrix*/
6:    Fixed[N] B /* Constants vector*/
7:    While i ≤ N Do
8:      B[i] ← Coeff[i][N]
9:      While j ≤ N-1 Do
10:       A[i][j] ← Coeff[i][j]
11:       j ← j + 1
12:     End While
13:     i ← i + 1
14:   End While
15:   i ← 0
16:   While i ≤ N Do
    /* Find pivot */
17:     Uint max ← i
18:     Uint k ← i + 1
19:     While i ≤ N Do
20:       If abs(A[k][i]) ≤ A[max][i]
21:         max = k
22:       End If
    /* swap row in matrix */
23:       A[i] ← A[max]
24:       A[max] ← temp
    /* swap corresponding values in constants matrix B */
25:       Fixed t ← B[i]
26:       B[i] ← A[max]
27:       B[max] ← temp
    /* pivot within A and B */
28:       j ← i + 1
29:       While j ≤ N Do
30:         Fixed factor ←A[j][i]/A[i][i]
31:         B[j] ← B[j] − (Factor * B[i])
32:         n ← i
33:         While n ≤ N Do
34:           A[j][n] ← A[j][n] − (Factor * A[i][n])
35:           n ← n + 1
36:         End While
37:         j ← j + 1
38:       End While
39:       k ← k + 1
40:     End While
41:   i ← i + 1
42:   End While
43: End Procedure
```

Algorithm 5 OPF Solver Smart contract: Solve row reduced DC-OPF system of linear equations

```
Structure: Solution
    Fixed: VoltageMagnitude
    Fixed: VoltagePhaseAngle
    Fixed: TimeStamp
End Structure
Address[3]: GenerationBuses
Mapping: Address → Solution: DC_OPF_Solution
Procedure Solve(Fixed[][] A, Fixed[] B) Returns (Bool)
    Uint N ← B.length
    Bool SystemSolved ← False
    Fixed [N] Solution
    i ← N-1
    While i ≥ N
        Fixedsolution ← 0
        j ← i + 1
        While j ≤ N
            sum ← sum + (A[i][j] * Solution[j])
            j ← i + 1
        End While
    solution[i] ← (B[i]-sum)/A[i][i]
    i ← ← i-1
    End While
    If (In solution, all Lagrange multipliers are positive and all constraints
    are satisfied) Then
        Solution  S
        SystemSolved ← True
        i ← 0
        While i ≤ 3
            S.VoltageMagnitude ← solution[i]
            S.VoltagePhaseAngle ← solution[i + 3]
            S.TimeStamp ← CurrentDate
            i ← i + 1
        End While
    End If
    Return SystemSolved
End Procedure
```

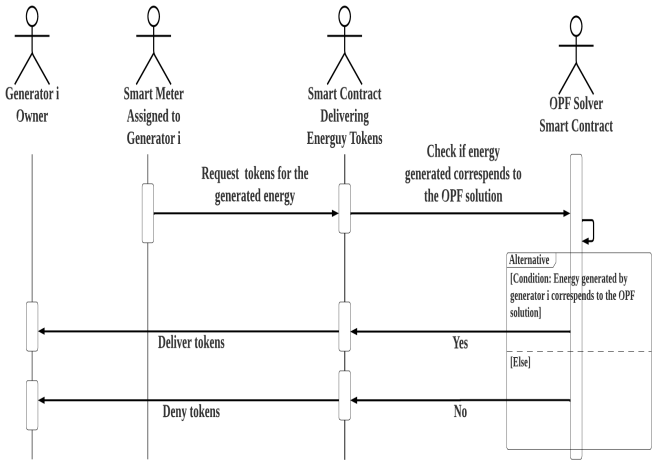

Figure 4. Flowchart illustrating how parties in the smart grid that do not comply with the OPF solution are not rewarded for their energy contribution.

The used example was solved for a daily (24 h) load distribution using day-ahead market load forecasting presented in [34] and depicted in Table 3.

Table 3. Hourly load power demand for 24 h.

Hour	P_D (MW)			Hour	P_D (MW)		
	D_1	D_2 & D_3	Total		D_1	D_2 & D_3	Total
1	132.66	44.22	221.1	13	153.06	51.02	255.1
2	122.4	40.8	204.0	14	149.64	49.88	249.4
3	115.62	38.54	192.7	15	147.96	49.32	246.6
4	112.2	37.4	187.0	16	147.96	49.32	246.6
5	108.84	36.28	181.4	17	154.74	51.58	257.9
6	110.52	36.84	184.2	18	170.04	56.68	283.4
7	112.2	37.4	187.0	19	163.26	54.42	272.1
8	119.04	39.68	198.4	20	161.52	53.84	269.2
9	136.02	45.34	226.7	21	159.84	53.28	266.4
10	149.64	49.88	249.4	22	165.42	52.14	260.7
11	153.06	51.02	255.1	23	147.96	49.32	246.6
12	154.74	51.58	257.9	24	137.76	45.92	229.6

The DC-OPF solution for the formulated problem for each hour of the day is presented in Table 4. It is of note that in Table 4 the $(^*)$ operator is used to denote an optimal entity and not the conjugate of the complex one.

Table 4. Three-bus example, DC-OPF solution.

Hour	P^*_{G1}	P^*_{G2}	P^*_{G3}	δ^*_2	δ^*_3
01	200.0	16.1	5.0	−0.0799	−0.1095
02	189.0	10.0	5.0	−0.0808	−0.1048
03	177.7	10.0	5.0	−0.0752	−0.0979
04	172.0	10.0	5.0	−0.0724	−0.0944
05	166.4	10.0	5.0	−0.0696	−0.0910
06	169.2	10.0	5.0	−0.0710	−0.0927
07	172.0	10.0	5.0	−0.0724	−0.0944
08	183.4	10.0	5.0	−0.0780	−0.1014
09	200.0	21.7	5.0	−0.0741	−0.1077
10	200.0	44.4	5.0	−0.0506	−0.1002
11	200.0	50.1	5.0	−0.0447	−0.0983
12	200.0	52.9	5.0	−0.0418	−0.0974
13	200.0	50.1	5.0	−0.0447	−0.0983
14	200.0	44.4	5.0	−0.0506	−0.1002
15	200.0	41.6	5.0	−0.0535	−0.1011
16	200.0	41.6	5.0	−0.0535	−0.1011
17	200.0	52.9	5.0	−0.0418	−0.0974
18	200.0	78.4	5.0	−0.0154	−0.0890
19	200.0	67.1	5.0	−0.0271	−0.0927
20	200.0	64.2	5.0	−0.0301	−0.0937
21	200.0	61.4	5.0	−0.0330	−0.0946
22	200.0	55.7	5.0	−0.0389	−0.0965
23	200.0	41.6	5.0	−0.0535	−0.1011
24	200.0	24.6	5.0	−0.0711	−0.1067

3.3. Execution Cost Results

Performing smart contract transactions in an Ethereum network requires a gas fee to be paid for each transaction. Gas refers to an abstract unit referring to the amount of Ethers (ETH) that need to be paid in accordance with the computational effort required to execute certain operations on the Ethereum network, this is because Ethereum transaction execution requires computational resource. Thus, issuing a transaction requires a fee to be

paid proportional to the transaction execution complexity. Another concept that is relevant to Ethereum transactions is the gas limit. When developing an Ethereum application that uses smart contracts, the cost of executing transactions is estimated using development tools such as Remix IDE and then a gas limit is set. Because smart contracts are deployed on the blockchain, they are immutable. Therefore, errors in smart contracts that have already been deployed cannot be fixed. Setting a gas limit prevents the smart contract from entering infinite loops that could be triggered by hackers who could exploit loopholes in the smart contract source code. Gas cost and gas limits in Ethereum applications are illustrated in the flowchart in Figure 5.

The gas cost of the on-chain OPF solution using the proposed system was evaluated using Remix IDE for the defined three-bus problem. The gas cost for execution was evaluated according to the value of the gas unit on 15 May 2020, which was 27.95 Gwei, as in [35], and the value of the Ethereum unit in dollars for the same day [36]. The hourly fluctuation of the gas price in dollars for the day is shown in Figure 6, accordingly the hourly gas cost of the on-chain OPF solution for the same selected date is shown in Figure 7.

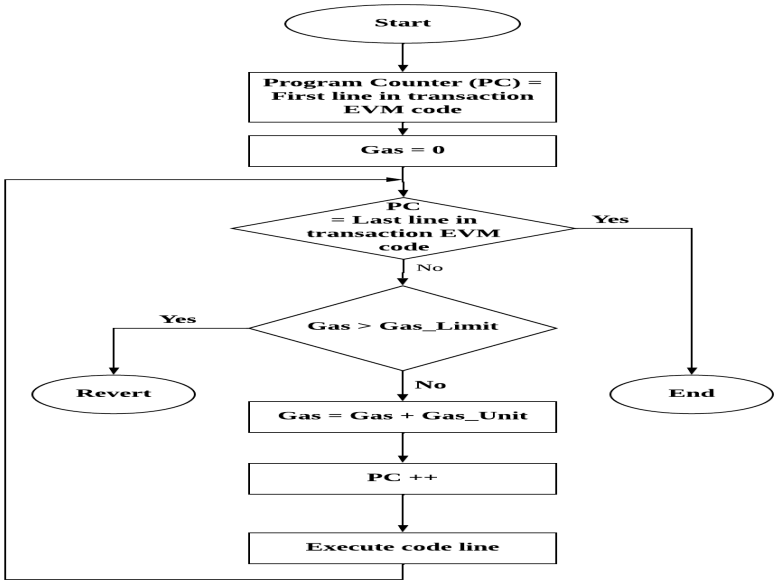

Figure 5. Gas consumption for Ethereum transaction execution.

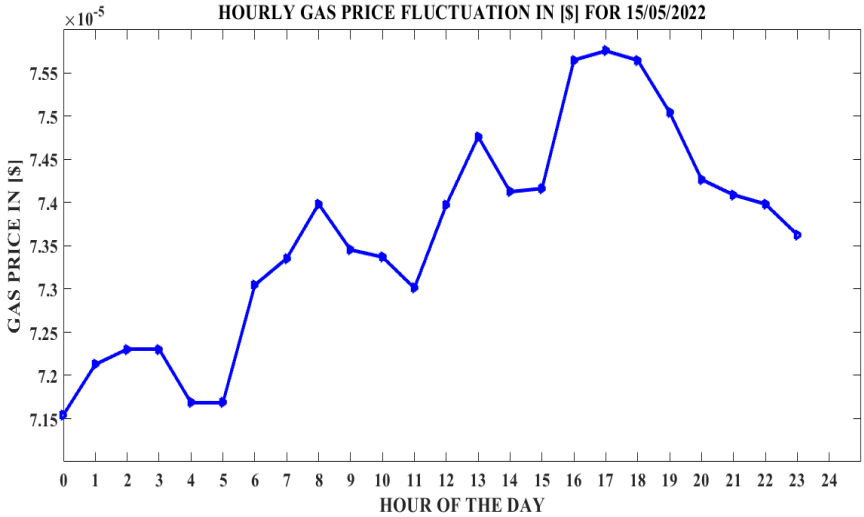

Figure 6. Hourly gas price fluctuation in US Dollars for 15 May 2022.

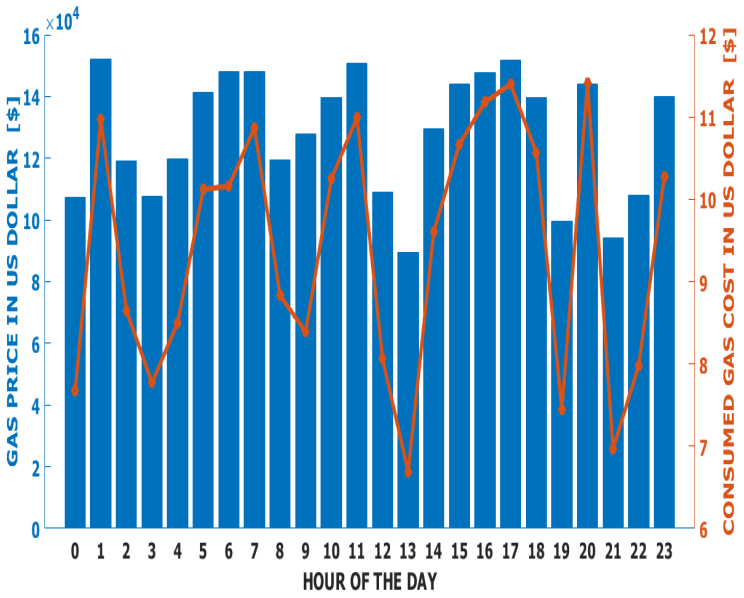

Figure 7. On-chain DC-OPF solution, execution cost in US Dollars, according to gas price fluctuation on 15 May 2022.

4. Enhanced Decentralized OPF Solving, Generalized for any Problem Formulated

As can be seen from the evaluation performed in the previous section, which showed very high execution costs for a simplified version of the OPF problem for a simple network, averaging about USD 9 per hour, such an approach is not conceivable for a much more complex problem such as the AC-OPF problem, which is non-convex and non-linear, and even less so for an even more complex problem that considers renewable energy sources with unpredictable generation capacity. The optimal solution is proposed by prover nodes in a decentralized and permission-less manner. Interested provers, can apply by depositing a defined amount of money into the smart contract, they are then notified by smart contract through an event when the bus loads are updated by the assigned unit. Provers solve the OPF problem off-chain, then send the solution they calculated, which represents the optimal combination of net power generated at each bus. Initially, the provers need to send their calculated solution in an encrypted form. They can only reveal their proposed solution after all eligible candidates have successfully committed their encrypted solution. After all of the proposed solutions are reveled and verified if they do correspond to the committed sent encryption by the smart contract, the smart contract then verifies if each proposed solution satisfies the problem constraints, then only the solutions giving the cheapest generation cost are considered. To clarify more, we assume M provers have successfully staked the defined required amount A to be eligible to propose a solution to the OPF problem. If $N, N \lesssim M$ provers form the M candidtates have proposed identical solutions satisfying the problem constraints and which is less expensive in term of generation cost than the solutions proposed by the rest of the $M - N$ provers. Then, the N provers will gain $\frac{(M-N)*A}{N}$ as a reward, where the rest of the $M - N$ provers lose their staked money, similar to a card betting game, as illustrated in Figure 8. In this case, the M total number of provers all propose an identical solution; it is adopted to control the generators if satisfying the problem constraints. In such a scenario, all the provers gain a symbolic reward to encourage them to continue solving the OPF problem.

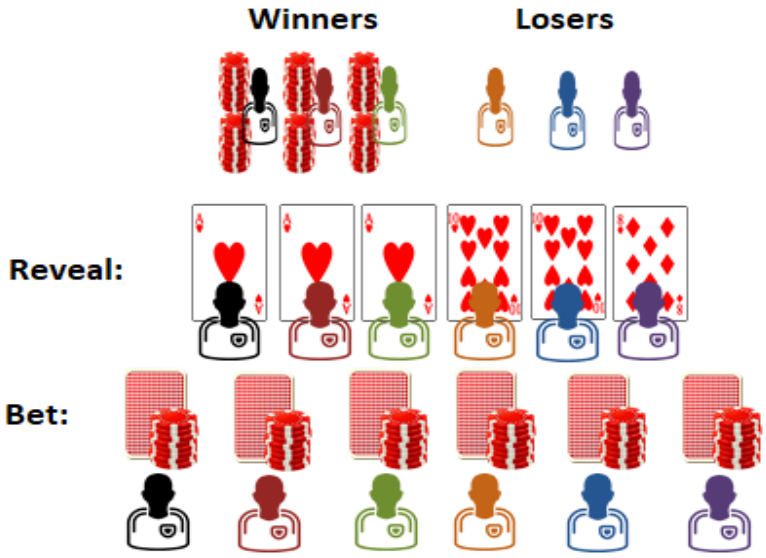

Figure 8. Proposed OPF solution scheme explained using a card betting game.

Since the provers proposed solution are only revealed after they are all committed to the blockchain, this gives rise to a Nash game scenario. To illustrate the game scenario applicable to the proposed platform, we assume two prover candidates proposing an OPF solution. The illustration with only two candidates can be extrapolated to n provers. In this game scenario, the two provers conspire to both propose identical non-optimal solution and gain the symbolic reward so that there is no loser. The two provers have two game strategies they can play. Either comply to the conspiracy and commit the agreed upon solution, or betray the second player and propose an optimal solution. We define the symbolic gain if both payers propose an identical solution to be 5, whereas the staked amount by each player to be 50. The gain for each player according to his played strategy is illustrated in Table 5.

Table 5. The gain for each player according to their played strategy.

Player1 \ Player2	Prpose Optimal Solution (Betray)	Collude
Propose optimal solution (Betray)	(+5,+5)	(+50,−50)
Collude	(−50,+50)	(+5,+5)

From Table 5, it can be seen that the total gain for each player when proposing an optimal solution is +55, whereas it is −45 if proposing a non-optimal one, which makes the proposed scheme immune against collusion. Such a system is suitable for any defined and formulated OPF problem where the OPF is calculated by impartial third parties who solve the OPF problem for remunerative purposes. Provers invest in developing the solution algorithms and performing the computations to find the most optimal solution and win the jackpot, similar to how miners invest in hashing power to be the first to solve the mining puzzle and win the mining reward. Since the OPF is computed off-chain and only the proposed solutions are compared and verified on-chain, this proposed system implies that much less heavy computation is performed on-chain without compromising the decentralization of the system.

4.1. Implementation for a 14-Bus AC-OPF Problem

In the AC model, transmission lines are represented by the π model. This model is the most commonly used representation of transmission lines in power networks and

is adopted by most AC-OPF simulation platforms, such as Matlab. The π model for the transmission line between two nodes i and j is shown in Figure 9.

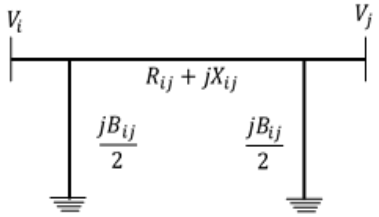

Figure 9. Transmission line π model.

In the π model, a transmission line consists of a series impedance $R_{ij} + jX_{ij}$ connected at the nodes i and j, and two equal shunt susceptances $y_{sh_i} = y_{sh_j} = j\frac{B_{ij}}{2}$. The line is mostly defined by its series admittance $y_{ij} = \frac{1}{R_{ij}+jX_{ij}}$ rather than its series impedance. Accordingly, as in the DC model, the OPF objective function aims to minimize the generation cost function $F(P_G)$, where $F(P_G) = \sum_{i=0}^{n} C(P_G)$. Thus, the objective function of the model AC-OPF is $MinF(P_G)$ and is subject to the constraints: AC flow, line current, generation active power, generation reactive power, voltage magnitude, and voltage angle defined in Equations (17)–(22), respectively.

$$\begin{aligned} &\bar{S_G} - \bar{S_L} = diag(\bar{V})\bar{Y}^*_{bus}\bar{Y}^* \\ &|\bar{Y}_{line,i\to j}\bar{V}| \le I_{line,max}, \end{aligned} \tag{17}$$

$$\begin{aligned} &|\bar{Y}_{line,i\to j}\bar{V}| \le I_{line,max} \\ &|\bar{Y}_{line,j\to}\bar{V}| \le I_{line,max}, \end{aligned} \tag{18}$$

$$0 \le P_G - P_{G,max}, \tag{19}$$

$$-Q_{G,max} \le Q_G - Q_{G,max}, \tag{20}$$

$$V_{min} \le V \le V_{max}, \tag{21}$$

$$\delta_{min} \le \delta \le \delta_{max}, \tag{22}$$

The $(^*)$ operator denotes the conjugate of a complex entity, the bar $(-)$ used above an entity is to express that it is a complex one; min, max indices are used to denote the maximum allowed value for an entity. In a network of N buses and M transmission lines, V is a vector of size N of voltages at each bus of the network. The $diag()$ operator is a vector to the diagonal matrix converter. In our case, $diag(V)$ returns an $N \times N$ diagonal matrix, where all non-diagonal elements starting from the left-upper to right-bottom are zeros and the diagonal elements are successively elements of the vector V. I_{line} is a vector of size M of the currents at each transmission of the network. Y_{line} is an $M \times N$ matrix corresponding to the line admittance matrix of the network, it links the bus voltages vector V to the lines current flow vector I_{line}, where $I_{line} = VY_{line}$. Because, the current flowing from a node i to a node j is different from the one flowing from node j to node i, there must be two different line admittance matrices Y_{line_1}, and Y_{line_2}. Accordingly, there are two I_{line} vectors, $I_{line_1} = \{I_{1\to 2}, I_{2\to 3}, ...I_{M\to M}\}$, $I_{line_2} = \{I_{1\leftarrow 2}, I_{2\leftarrow 3}, ...I_{M\leftarrow M}\}$, where $I_{line_1} = VY_{line_1}$ and $I_{line_2} = VY_{line_2}$. A row k in the line admittance matrix corresponds to the transmission line with current I_k, which is the kth element in the linked line current flow vector I_{line}, given that row k corresponds to line $i \to j$, the ith element in the row $Y_{line_{ki}} = y_{sh_i} + y_{ij}$, whereas the jth element in the row $Y_{line_{kj}} = -y_{ij}$, the rest of the row elements are all null. Y_{bus} is an $N \times N$ matrix, representing the bus admittance matrix. Diagonal element $Y_{bus_{ii}}$ of the matrix corresponding to a row i, are the sum of the shunt susceptance at bus i and the sum of all series admittance of all the lines connected to bus i. For non-diagonal

elements $Y_{bus_{ij}, i \neq j}$, they are either null if there is no connecting line between bus i and bus j, otherwise $Y_{bus_{ij}} = -y_{ij}$. S represents a vector of size N of the total net apparent power at each bus of the network. Given the $Y_{line, i \to j}$ matrix, $S = S_G - S_L$, where S_G and S_L are both vectors of size N. S_G is the total net generation power vector and S_L is the net load demand power vector. S has two components. A real component representing the active power flow $P_G - P_L = Re(S_G - S_L)$, and an imaginary one representing the reactive power flow $Q_G - Q_L = Im(S_G - S_L)$. P_G, Q_G, P_L, Q_L, are the vectors of size N of the active and reactive power generation at each bus and the active and reactive load demand power at each bus as well, respectively. Both the active and reactive power at the given node i are represented in Equation (23), with respect to both the bus active and reactive load demand power, respectively.

$$\begin{cases} P_{Gi} = \sum_{j=1}^{N} |V_i||V_j|(G_{ij}cos(\theta_k - \theta_j) + B_{kj}sin(\theta_i - \theta_j)) + P_{Li}, \\ Q_{Gi} = \sum_{j=1}^{N} |V_i||V_j|(G_{ij}sin(\theta_k - \theta_j) - B_{kj}cos(\theta_i - \theta_j)) + Q_{Li} \end{cases} \tag{23}$$

G_{ij} and B_{ij} represent the real and imaginary parts of Y_{ij} in the admittance matrix Y that should be preset in the smart contract. Therefore, for each bus, provers must specify the voltage magnitude and phase at the bus, and the voltage magnitudes and phases at each bus connected to that bus. P_{Gi} and Q_{Gi} are accordingly calculated within the smart contract using sin and cos functions available in the smart contract in [37]. Thus, provers should only upload for each bus the respective bus voltage magnitudes and phase angles corresponding to the bus, both the power active and reactive generation, in Equation (23). The pseudocode in Algorithm 6 illustrates how the interested prover candidates apply to solve the OPF problem by depositing the required amount. As demonstrated applications are no more accepted once the preset number of maximum applications is reached.

Algorithm 6 OPF Smart Contract: Apply

```
Uint: SmartContractBalance
Uint: DepositRequiredAmount
Uint: TotalNumberOfPermittedProvers
Uint: NumberOfProvers ← 0
Address:[] OPFproverCandidates
Mapping: Address → Uint:  ProversBalance
Procedure Payable Apply()
Uint: NumberOfProvers ← NumberOfProvers + 1
   If msg.value ≤ DepositRequiredAmount
      Revert()
   Else If NumberOfProvers≤ TotalNumberOfPermittedProvers
      ProverBlance[msg.sender] ← ProverBlance[msg.sender] + msg.value
      OPFproverCandidates.Push(msg.value)
      SmartContractBalance ← SmartContractBalance + msg.value
   Else
      Revert()
   End If
End Procedure
```

Before provers start uploading their proposed solution to the smart contract, the network buses' load demand is updated by the assigned unit on an hourly basis, the smart contract date is also updated on the same occasion, similar to the model described in the previous section. This is illustrated in pseudocode in Algorithm 7. It is of note that in order to not make the pseudocodes excessively long, only active generation and demand power are considered in the presented pseudocodes. However, the system constraints related

to buses reactive power are considered during the implementation used to evaluate the execution cost of the model.

Algorithm 7 OPF Smart Contract: UpdateBusesLoadPowerDemand

```
Address: LoadMonitoringUnit
Fixed[14]: BusesLoadPowerDemand
Fixed[14][14]: BusAdmittanceMatrixReal
Fixed[14][14]: BusAdmittanceMatrixImag
Procedure UpdateBusesLoadPowerDemand(Fixed[14] BusesLoadP)
  If msg.sender != LoadMonitoringUnit Then
    Revert()
  Else
    For all L in BusesLoadP Do
      BusesLoadPowerDemand.Push(L)
    End For
    UpdateCurrentDate()
    Emit Event: NotifyProverCandidates()
  End If
End Procedure
```

As explained earlier, the provers first upload an encrypted solution, which they cannot reveal until all provers have uploaded an encrypted solution. Once this is done, the smart contract sends out an event to notify the provers to reveal their solution, which is then checked to see if it matches the corresponding encrypted message sent earlier. This is explained in pseudocode in Algorithm 8.

After each prover discloses their proposed solution, it is checked to see if it satisfies the problem constraints. If it does, the corresponding generation cost is calculated and assigned to the respective prover. This is described in the pseudocode of the Algorithm 9.

Finally, all valid solutions are compared based on their calculated generation costs, and then the prover that proposed the optimal solution is rewarded, as described in the pseudocode in Algorithm 10. The entity relationship diagram for the proposed smart contract-based solution is shown in the following Figure 10.

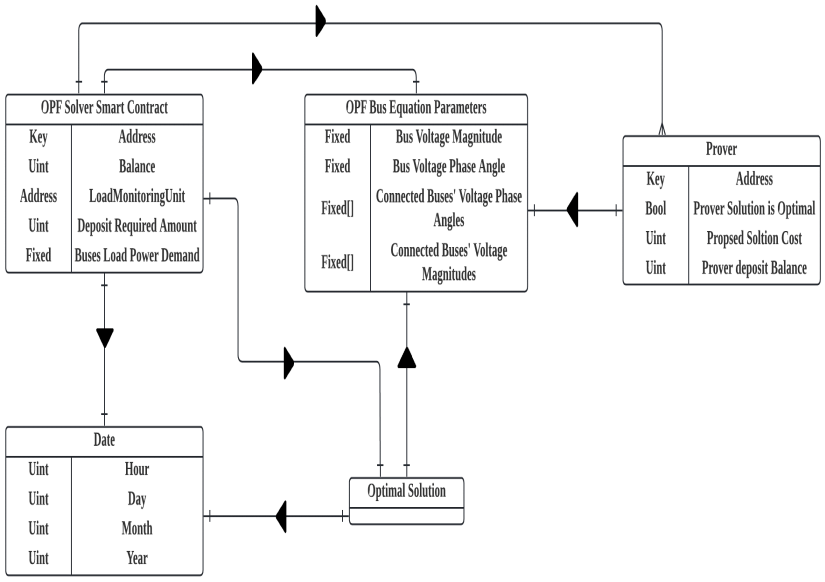

Figure 10. Entity-relationship diagram for the enhanced OPF solution model.

Algorithm 8 OPF Smart Contract: Send encrypted solution/reveal solution

```
Structure: BusGeneratedPowerFlowEquationParameters
    Fixed[] ConnectedBusesPhaseAngles
    Fixed[] ConnectedBusesVoltageMagnitude
    Fixed BusPhaseAngle
    Fixed BusVoltageMagnitude
End Structure:
Uint EncryptedSolutionsCommited ← 0
Uint SolutionsRevealed ← 0
Mapping: Address → BusGeneratedPowerFlow-
EquationParameters[]: ProverPoposedSolution
Fixed[14][14]: BusAdmittanceMatrixImagComponents
Mapping: Address → Bytes32: ProverEncryptedSolution
Procedure SendEncryptedSolution(Bytes32 EncryptedSolution)
    If  ProversBalance[msg.sender]≤ DepositRequiredAmount  Then
        Revert()
    Else If EncryptedSolutionCommited= TotalNumberOfPermittedProvers
        Emit Event: NotifyProversToRevealSolution
    Else
        ProverEncryptedSolution(msg.sender) ← EncryptedSolution
        EncryptedSolutionsCommited ← EncryptedSolutionsCommited + 1
    End If
End Procedure
Procedure Reveal-OPF-Solution(BusGeneratedPower-
FlowEquationParameters[] Eq)
    Uint a ← EncryptedSolutionsCommited
    Uint b←TotalNumberOfPermittedProvers
    Uint c ← ProversBalance[msg.sender]
    Uint d ← DepositRequiredAmount
    If  a ≤ b || c≤ d Then
        Revert()
    Else If SolutionsRevealed ≥ b
        ProversSolutionsCost()
        RewardOPFsolvers()

    Else
        Bytes32 c ← ProverEncryptedSolution(msg.sender)
        Bytes32 d←keccak256(abi.encode(Eq))
        If  c != d  Then
            Revert()
        Else
            ProverPoposedSolution(msg.sender) ← Eq
            SolutionsRevealed ← SolutionsRevealed + 1
        End If
    End If
End Procedure
```

Algorithm 9 OPF Smart Contract: ProversSolutionCost

Mapping: Address → **Bool:** ProverSolutionIsOptimal Distances
Mapping: Address → **Fixed:** ProposedSolutionCost Distances
Procedure ProverSolutionsCost()
  Fixed: GenerationCost ← 0
  For Each P **in** OPFproverCandidates **do**
    Uint: i ← 0
    Uint: j ← 0
    Fixed θ_i ← ProverProposedSolution[i].BusPhaseAngle
    Fixed V_i ← ProverProposedSolution[i].BusVoltageMagnitude
    Fixed[] θ_j ← ProverProposedSolution[i].ConnectedBusesPhaseAngles
    Fixed[] V_j ← ProverProposedSolution[i].ConnectedBusesVoltageMagnitudes
    Fixed Pg_i ← 0
    Fixed G_{ij} ← BusAdmittancesMatrixReal[i][j]
    Fixed B_{ij} ← BusAdmittancesMatrixImag[i][j]
    Fixed Pd_i ← BusesLoadPowerDemand[i]
    while i ≤ 14 **do**
      while j ≤ 14 **do**
        $Pg_i \leftarrow Pg_i + (V_j * Vj(G_{ij} * cos(\theta_i - \theta_j) - B_{ij} sin(\theta_i - \theta_j)) + Pd_i)$
        GenerationCost ← GenerationCost + GenerationCostFunction(Pg_i)
        j ← j + 1
      End while
      i ← i + 1
    End while
    ProposedSolutionCost(P) ← GenerationCost
  End For
End Procedure

4.2. Execution Cost Results

The proposed scheme was used to solve the AC-OPF problem for a 14-bus. The model used was an IEEE 14-bus system. All network parameters can be found in [38]. The execution cost of the proposed system was evaluated using the Remix IDE development tool. It enables the execution of the developed smart contract and logs the execution cost of the smart contract functions in gas abstraction. Although this provides insight into the execution cost of the smart contract functions, testing the model in a deployed private blockchain network would provide a more accurate assessment, as the development tools do not account for non-deterministic technical variables. Three evaluations were performed with 20, 50, and 100 provers, and a single optimal solution was proposed. The obtained results in term of execution cost in US dollars, according to gas price fluctuation on 15 May 2022, are illustrated in Figure 11.

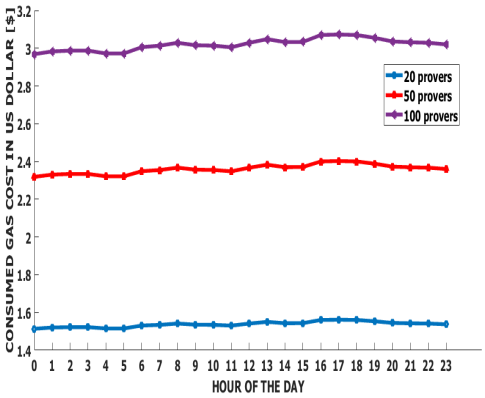

Figure 11. Execution cost of AC-OPF solution, cost in US dollars, depending on gas price fluctuation on 15 May 2022, using the proposed scheme.

Algorithm 10 OPF Smart Contract: RewardOPFsolver

```
1:  OPF-BusesEquationParameters[] OptimalSolution
2:  Procedure RewardOPFsolver()
3:    Address P ←OPFproverCandidates[0]
4:    Uint OptimalCost ← ProposedSolutionCost(P)
5:    Uint i ← 1
6:    While i ≤ OPFproverCandidates.length()  Do
7:      P ← OPFproverCandidates[i]
8:      If ProposedSolutionCost (P)≤ OptimalCost Then
9:        OptimalCost ←ProposedSolutionCost(P)
10:     End If
11:     i ← i + 1
12:   End While
13:   i ← 0
14:   Uint NumberOfWinners ← 0
15:   While i ≤ OPFproverCandidates.length() Do
16:     P ← OPFproverCandidates[i]
17:     If ProposedSolutionCost(P) == OptimalCost Then
18:       OptimalSolution ← ProverProposedSolution(P)
19:       ProverSolutionIsOptimal(P) ←True
20:       NumberOfWinners ← NumberOfWinners + 1
21:     Else
22:       ProverSolutionIsOptimal(P) ← False
23:     End If
24:     i ← i + 1
25:   End While
26:   PriceMoney ← SmartContractBalance/NumberOfWiners
27:   i ← 0
28:   While i ≤ OPFproverCandidates.length() Do
29:     P ← OPFproverCandidates[i]
30:     If ProposedSolutionIsOptimal(P) = True Then
31:       P.transfer(PriceMoney)
32:       ProversDepositBlance(P) ← 0
33:     Else
34:       ProversDepositBlance(P) ← 0
35:     End If
36:     i ← i + 1
37:     SmartContractBalance ← 0
38:   End While
39: End Procedure
```

5. Discussion

From the execution cost results for the two schemes presented in Sections 3 and 4, shown in Figure 12, it can be seen that the extended generalized scheme presented in Section 4 provides a much cheaper execution for a much more complex problem compared to the scheme presented in Section 3, without compromising the decentralization of the system. It can also be seen that the execution cost is much more stable over the daytime hours when using the enhanced model, while the execution cost shows a much more fluctuating pattern when solving the OPF problem on-chain. This is due to the fact that OPF is an NP problem and the number of iterations required to solve a given OPF problem is not deterministic. However, it can be seen that increasing the number of provers in the second scheme increases the execution cost, since the smart contract must loop through all provers and verify and compare all proposed results to determine the provers with the optimal solution. The number of provers allowed to participate in an OPF solution loop must be carefully selected before it is defined and fixed in the smart contract. On the other hand, this solution can still be considered expensive, as the operation can reach a few dollars per hour for very complex problems. However, we believe that this is still a better solution than a centralized solution that requires the use of hardware. Such an approach has two drawbacks. First, it does not go in the direction of keeping P2P energy trading platforms as decentralized as possible, and second, it is not necessarily a cheaper solution because it requires the use of technical staff for ongoing maintenance and monitoring in addition to the high deployment costs, which are one-time fixed investment costs. Figure 13 shows the minimum hourly wage for various industrialized countries [39].

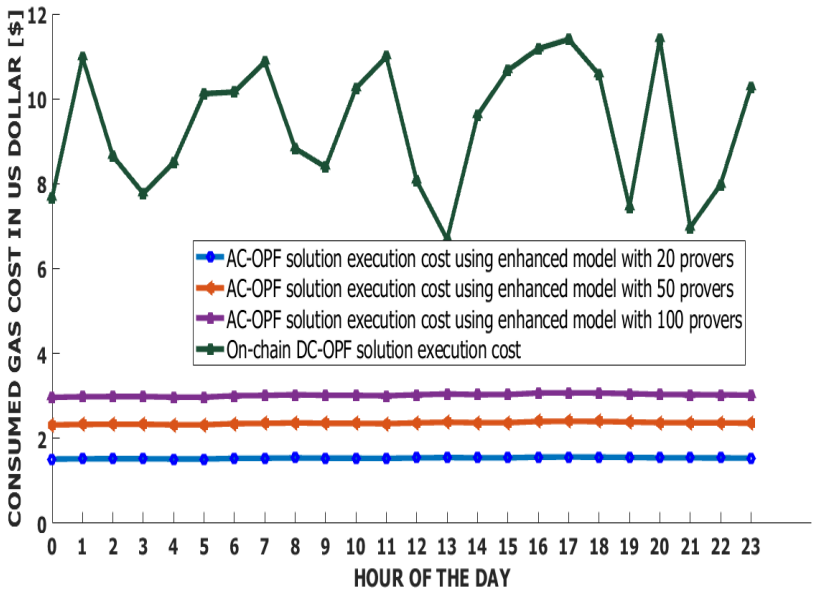

Figure 12. Comparison of execution costs between on-chain OPF solutions for a 3-bus DC-OPF problem and the solution for a 14-bus AC-OPF problem using the enhanced proposed model.

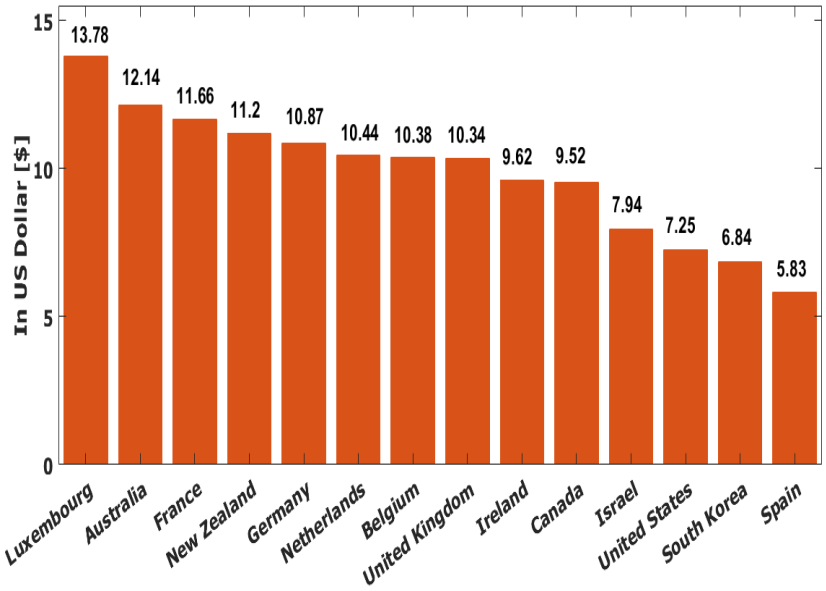

Figure 13. Hourly minimum wage for different industrialized countries (2022).

6. Conclusions

In this paper, we present a blockchain-based demand-side management for smart grids and P2P energy trading platforms using an OPF control system based on smart contracts. In the proposed system, no centralized entities are entrusted with generation control of the various energy sources in the grid, and no control hardware deployed by the DSO is required. Rather, the various generators and owners of energy sources are forced by a decentralized consortium to comply with the calculated OPF solution, and those who do not comply would see their energy contribution wasted. The first approach, where the OPF is fully calculated by a smart contract, showed very high execution costs for a simplified linear approach to the OPF problem, which led us to conclude that although this is a fully decentralized solution, it is not realistic due to the complexity of the OPF problem. Therefore, a different approach was taken in which the OPF problem is not computed on-chain. Instead, different solutions proposed by different provers are compared and verified through a smart contract, considering only the valid and cost-optimal proposals and rewarding those who proposed them. The proposed system was designed as a Nash game. The fewer provers proposed an optimal solution, the more a particular prover with an optimal solution proposal wins. This prevents both collusion among provers and laziness in computing the OPF solution, as they risk losing their invested money if their proposal is not optimal compared to the others. This system was evaluated for solving a non-convex, non-linear AC-OPF problem for a 14-bus network and showed reasonable execution costs that increased with the number of participating provers. Apart from being a decentralized solution with no central authority controlling the power flow, the proposed system can even be considered as a lower cost solution considering the high cost of hardware deployment and staff wages for continuous maintenance and monitoring in most industrialized countries.

Author Contributions: Conceptualization, Y.M. and M.H.H.; methodology, Y.M., M.H.H. and M.R.I.; software, Y.M.; validation, T.S.G., M.H.H. and M.R.I.; investigation, Y.M. and M.H.H.; resources, S.F.T. and M.M.; writing—original draft preparation, Y.M.; writing—review and editing, M.H.H. and M.R.I.; visualization, R.A.; supervision, M.H.H., M.R.I. and T.S.G.; project administration, M.M.; funding acquisition, S.F.T. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based upon work supported by the Air Force Office of Scientific Research under award number FA2386-21-1-4046. The research work was conducted at the IoT and wireless communication protocols laboratory, International Islamic University Malaysia (IIUM). Yacine Merrad is grateful to IIUM Tuition Fee Waiver program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, Z.; Liu, F.; Ma, Z.; Chen, Y.; Jia, M.; Wei, W.; Wu, Q. Distributed generalized nash equilibrium seeking for energy sharing games in prosumers. *IEEE Trans. Power Syst.* **2021**, *36*, 3973–3986. [CrossRef]
2. Sotkiewicz, P.M.; Vignolo, J.M. Towards a cost causation-based tariff for distribution networks with DG. *IEEE Trans. Power Syst.* **2007**, 22, 1051–1060. [CrossRef]
3. Rinaldi, G.; Menon, P.P.; Edwards, C.; Ferrara, A. Higher order sliding mode observers in power grids with traditional and renewable sources. *IEEE Control. Syst. Lett.* **2019**, *4*, 223–228. [CrossRef]
4. Wu, X.; Conejo, A.J. Distribution Market Including Prosumers: An Equilibrium Analysis. *IEEE Trans. Smart Grid* **2022**. [CrossRef]
5. Merrad, Y.; Habaebi, M.H.; Islam, M.R.; Gunawan, T.S.; Elsheikh, E.A.; Suliman, F.M.; Mesri, M. Machine Learning-Blockchain Based Autonomic Peer-to-Peer Energy Trading System. *Appl. Sci.* **2022**, *12*, 3507. [CrossRef]
6. Junlakarn, S.; Kokchang, P.; Audomvongseree, K. Drivers and Challenges of Peer-to-Peer Energy Trading Development in Thailand. *Energies* **2022**, *15*, 1229. [CrossRef]
7. Shabani, M. Blockchain-based platforms for genomic data sharing: A decentralized approach in response to the governance problems? *J. Am. Med. Inform. Assoc.* **2019**, 26, 76–80. [CrossRef]
8. Han, D.; Zhang, C.; Ping, J.; Yan, Z. Smart contract architecture for decentralized energy trading and management based on blockchains. *Energy* **2020**, *199*, 117417. [CrossRef]
9. Gajić, D.B.; Petrović, V.B.; Horvat, N.; Dragan, D.; Stanisavljević, A.; Katić, V.; Popović, J. A Distributed Ledger-Based Automated Marketplace for the Decentralized Trading of Renewable Energy in Smart Grids. *Energies* **2022**, *15*, 2121. [CrossRef]
10. Department of Energy and Climate Change (DECC). *Community Energy Strategy: Full Report*; DECC: London, UK, 2014.
11. Afzal, M.; Huang, Q.; Amin, W.; Umer, K.; Raza, A.; Naeem, M. Blockchain enabled distributed demand side management in community energy system with smart homes. *IEEE Access* **2020**, *8*, 37428–37439. [CrossRef]
12. Karandikar, N.; Chakravorty, A.; Rong, C. Blockchain based transaction system with fungible and non-fungible tokens for a community-based energy infrastructure. *Sensors* **2021**, 21, 3822. [CrossRef] [PubMed]
13. Jayachandran, M.; Rao, K.P.; Gatla, R.K.; Kalaivani, C.; Kalaiarasy, C.; Logasabarirajan, C. Operational concerns and solutions in smart electricity distribution systems. *Util. Policy* **2022**, *74*, 101329. [CrossRef]
14. Wang, Z.; Anderson, C.L. A progressive period optimal power flow for systems with high penetration of variable renewable energy sources. *Energies* **2021**, *14*, 2815. [CrossRef]
15. Riaz, M.; Hanif, A.; Masood, H.; Khan, M.A.; Afaq, K.; Kang, B.G.; Nam, Y. An Optimal Power Flow Solution of a System Integrated with Renewable Sources Using a Hybrid Optimizer. *Sustainability* **2021**, *13*, 13382. [CrossRef]
16. Nusair, K.; Alasali, F. Optimal power flow management system for a power network with stochastic renewable energy resources using golden ratio optimization method. *Energies* **2020**, *13*, 3671. [CrossRef]
17. Hassan, M.H.; Kamel, S.; Selim, A.; Khurshaid, T.; Domínguez-García, J.L. A modified Rao-2 algorithm for optimal power flow incorporating renewable energy sources. *Mathematics* **2021**, *9*, 1532. [CrossRef]
18. Jamil, M.; Mittal, S. Hourly load shifting approach for demand side management in smart grid using grasshopper optimisation algorithm. *IET Gener. Transm. Distrib.* **2020**, *14*, 808–815. [CrossRef]
19. Vale, Z.; Faria, P.; Abrishambaf, O.; Gomes, L.; Pinto, T. MARTINE—A Platform for Real-Time Energy Management in Smart Grids. *Energies* **2021**, *14*, 1820. [CrossRef]
20. Ahmad, A.; Javaid, N.; Mateen, A.; Awais, M.; Khan, Z.A. Short-term load forecasting in smart grids: An intelligent modular approach. *Energies* **2019**, *12*, 164. [CrossRef]
21. Esmat, A.; de Vos, M.; Ghiassi-Farrokhfal, Y.; Palensky, P.; Epema, D. A novel decentralized platform for peer-to-peer energy trading market with blockchain technology. *Appl. Energy* **2021**, *282*, 116123. [CrossRef]
22. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, W.; Chen, X.; Weng, J.; Imran, M. An overview on smart contracts: Challenges, advances and platforms. *Future Gener. Comput. Syst.* **2020**, *105*, 475–491. [CrossRef]
23. Khan, S.N.; Loukil, F.; Ghedira-Guegan, C.; Benkhelifa, E.; Bani-Hani, A. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-Peer Netw. Appl.* **2021**, 14, 2901–2925. [CrossRef] [PubMed]
24. Jansen, M.; Hdhili, F.; Gouiaa, R.; Qasem, Z. Do smart contract languages need to be turing complete? In Proceedings of the InInternational Congress on Blockchain and Applications, Avila, Spain, 26–28 June 2019; pp. 19–26.
25. Yang, Z.; Zhong, H.; Xia, Q.; Kang, C. Fundamental review of the OPF problem: Challenges, solutions, and state-of-the-art algorithms. *J. Energy Eng.* **2018**, *144*, 04017075.

26. Javed, H.; Muqeet, H.A.; Irfan, M. Recent Trends, Challenges and Future Aspects of P2P Energy Trading platforms in Electrical-based Networks Considering Blockchain Technology: A Roadmap Towards Environmental Sustainability. *Front. Energy Res.* **2022**, *10*, 134. [CrossRef]
27. Wang, G.; Wang, S.; Bagaria, V.; Tse, D.; Viswanath, P. Prism removes consensus bottleneck for smart contracts. In Proceedings of the 2020 Crypto Valley Conference on Blockchain Technology (CVCBT), Rotkreuz, Switzerland, 11–12 June 2020; pp. 68–77.
28. Du, Y.; Wang, Z.; Leung, V. Blockchain-enabled edge intelligence for IoT: Background, emerging trends and open issues. *Future Int.* **2021**, *13*, 48. [CrossRef]
29. Xing, C.; Chen, Z.; Chen, L.; Guo, X.; Zheng, Z.; Li, J. A new scheme of vulnerability analysis in smart contract with machine learning. *Wirel. Netw.* **2020**, *8*, 1–10. [CrossRef]
30. Christie, R. Power Systems Test Case Archive. August 1993. Available online: http://www.ee.washington.edu/research/pstca/pf30/pg_tca30bus.htm (accessed on 16 June 2022).
31. Ergun, H.; Dave, J.; Van Hertem, D.; Geth, F. Optimal power flow for AC–DC grids: Formulation, convex relaxation, linear approximation, and implementation. *IEEE Trans. Power Syst.* **2019**, *34*, 2980–2990. [CrossRef]
32. Durvasulu, V.; Hansen, T.M. Market-based generator cost functions for power system test cases. *IET-Cyber-Phys. Syst. Theory Appl.* **2018**, *3*, 194–205. [CrossRef]
33. Gordon, G.; Tibshirani, R. Karush-kuhn-tucker conditions. *Optimization* **2012**, *10*, 725.
34. Shahidehpour, M.; Yamin, H.; Li, Z. *Market Overview in Electric Power Systems*; Wiley-IEEE Press: Hoboken, NJ, USA, 2002; pp. 1–20.
35. Ethereum Average Gas Price. Available online: https://ycharts.com/indicators/ethereum_average_gas_price (accessed on 16 June 2022).
36. Ethereum Price. Available online: https://ycharts.com/indicators/ethereum_price (accessed on 16 June 2022).
37. Sikora, Basic Trigonometry Functions Smart Contract. Available online: https://github.com/Sikorkaio/sikorka/blob/master/contracts/trigonometry.sol (accessed on 16 June 2022).
38. Christi, R. Power Systems Test Case Archive. August 1993. Available online: http://labs.ece.uw.edu/pstca/pf14/pg_tca14bus.htm (accessed on 16 June 2022).
39. Minimum Wage by Country. 2022. Available online:https://worldpopulationreview.com/country-rankings/minimum-wage-by-country (accessed on 16 June 2022).