Article

Hardware Implementation of a Home Energy Management System Using Remodeled Sperm Swarm Optimization (RMSSO) Algorithm

Senthil Prabu Ramalingam and Prabhakar Karthikeyan Shanmugam *

School of Electrical Engineering, Vellore Institute of Technology, Vellore 632014, Tamil Nadu, India; rsenthilprabu77@gmail.com
* Correspondence: sprabhakarkarthikeya@vit.ac.in; Tel.: +91-9894610689

Abstract: A remodeled sperm swarm optimization (RMSSO) algorithm for a home energy management (HEM) system is proposed, and its real-time efficacy was evaluated using a hardware experimental model. This home environment comprised sixteen residential loads, a smart meter and a Raspberry Pi controller to optimize the energy consumption cost (ECC) in response to the Indian day-ahead pricing (DAP) scheme. A wired/wireless communication network was considered to communicate with the smart meter and controller. To address this optimization problem, the sperm swarm optimization (SSO) algorithm’s constriction coefficient was remodeled to improve its global searching capability and proposed as RMSSO. For the first time, salp swarm optimization (SSA), SSO, and RMSSO algorithms were employed to schedule home appliances in the Indian scenario. To validate the proposed technique’s outcome, the results were compared to those of the conventional SSO and SSA algorithms. This problem was solved using the Python/GUROBI optimizer tool. As a consequence, consumers can use this control strategy in real-time to reduce energy consumption costs.

Keywords: home energy management system; day-ahead pricing; constriction factor; remodeled sperm swarm optimization; salp swarm optimization; sperm swarm optimization; user satisfaction

1. Introduction

The climatic condition and the advancement of technology are the motivating factors behind energy management in residential buildings. In developing countries like India, electricity consumption is increasing in tandem with the country’s economic development. It is expected to rise by 140% by 2021–2022, with Tamil Nadu, Telangana, and Karnataka accounting for nearly 80% of the increase [1]. As a result, making optimal use of power by consumers is a key step in reducing energy demand growth. An energy management control unit (EMU) is used in residential buildings to make sure that energy is used efficiently. This is done by properly monitoring, regulating, and optimizing energy usage.

Nowadays, the conventional electrical grid has been transformed into a smart grid (SG). The European Union Commission Task Force for Smart Grids has provided the smart grid definition as follows: “SG is an electricity network that can cost efficiently integrate the behavior and actions of all users connected to it—generators, consumers and those that do both—in order to ensure economically efficient, sustainable power system with low losses and high levels of quality, security, of supply and safety.” Thus, the SG reduces energy waste and consumption costs and increases reliability, efficiency, and transparency of the energy supply. In the development of SG, demand-side management (DSM) is considered as an important feature in providing economic benefits to the consumer through controlling, monitoring, protecting, and optimizing the home appliance operation. DSM is also aimed at benefiting the utility control center by reducing the stress during peak hours. The utility control center implements the demand response (DR) program by bringing consumers into the picture in the process of energy management program. The DR can
be divided into three categories: price-based or rate-based DR programs, incentive- or event-based DR programs, and demand-reduction-bid-based DR programs. This proposed work follows price-based DR programs that include real-time pricing (RTP), day-ahead pricing (DAP), time-of-use (TOU) pricing, and critical peak pricing (CPP) programs. These pricing schemes have played a vital role in attaining the monetary benefit for the smart home consumers [2–5].

Typically, a smart home is a part of smart grid (SG) and is defined as “A smart home is a residence incorporating a communications network between electric household appliances and services” [6]. A smart home consists of a real-time monitoring system that communicates with each device to optimize energy use. Optimization of energy consumption cost can be accomplished via stochastic optimization approaches with accurate probabilistic parameter estimation. Thus, real-time home energy optimization is the ideal solution, even when energy demand and consumption costs fluctuate.

Most of the research was carried out on optimization techniques to address the energy management problems for residential users: linear programming (LP), integer linear programming (ILP), mixed-integer programming (MIP), non-convex programming, mixed-integer linear programming (MILP), and non-deterministic polynomial-time hardness (NP-hardness) techniques. However, the computational time for these optimization algorithms is prohibitively long. On the other hand, evolutionary algorithms provide a fast and near-optimal solution to these problems [7–9].

In smart homes, the utility control center manually performs load shifting or sheds a particular load for a certain period of time through the existing electricity system to minimize peak formation during peak hours [10,11]. As a result, only the consumers benefit from such actions, not the utility control center. Furthermore, moving the load from peak to off-peak periods lowers peak demand and energy consumption costs, but it still outrages the user’s satisfaction level. It should be mentioned here that there is always a trade-off between the user satisfaction level and energy consumption cost, and achieving both concurrently is the most difficult task [12]. Thus, to achieve them together, some of the major constraints like daily energy consumption (kW), peak-average ratio (PAR), energy price signals, and user satisfaction have to be considered. As a consequence of these challenges, effective energy management algorithms that can handle all sorts of loads and adapt to the uncertainties of energy prices are required [13].

In this regard, the authors of [14,15] have developed scheduling algorithms based on consumption cost reduction and consumer preference to manage residential appliances, which achieve the desired trade-off between economic benefits and consumer preference. Similarly, machine learning techniques, linear and dynamic programming, particle swarm optimization (PSO), fuzzy methods, and game theory are among the optimization techniques used in home energy management systems to schedule and control home appliances to provide economic benefits to consumers [16–21]. However, consumers are still not able to attain both user satisfaction and cost savings together, which are the drawbacks of the existing DR programs for DSM.

Recent literature suggests that home appliances can be categorized based on their operational behavior and energy consumption pattern as non-schedulable, schedulable, and controllable appliances to maximize the consumer satisfaction level and to attain the flexibility of scheduling [22–25]. The authors of [26] have presented the definition of energy management as a set of strategies and functions that can optimize energy use. These sets of strategies effectively balance the demand and supply. Energy management is the process of monitoring, controlling, and optimizing the energy usage in residential buildings. It efficiently optimizes energy consumption costs and minimizes the peak-average ratio.

The authors of [27,28] have suggested that the Harris Hawks Optimization (HHO) algorithm and the Water Cycle Algorithm (WCA) effectively minimize the overall power losses and maximize the load balance at the distribution network level. The authors of the Harris Hawks Optimization algorithm and the Water Cycle Algorithm have compared them with particle search optimization (PSO), the harmony search algorithm (HSA), the
fireworks algorithm (FWA), the Cuckoo search algorithm (CSA), and the uniform-voltage-distribution-based constructive algorithm (UVDA). The authors claim that their algorithms are the best at improving the efficiency and sustainability of the distribution grid.

In [29], a hybrid optimization algorithm predicts the PV power generation by combining a convolutional neural network (CNN) and the salp swarm algorithm (SSA). This forecast is based on the weather (rainy, heavy cloudy, moderately cloudy, lightly cloud, and sunny). The CNN is applied to predict the next day’s weather type, and the SSA technique is used to optimize each model. Thus, to enhance the SSA technique’s exploring and exploiting capabilities, a simulated annealing mechanism is employed, which is based on symmetric perturbation for automated compliance checking in residential microgrids [30]. Therefore, residential microgrids and smart homes require an effective energy management unit that is capable enough to forecast and solve the microgrids’ problems in advance and provide the ideal solution to balance the demand and supply. In [31], the authors have proposed a rainfall algorithm with TOU pricing to schedule the home appliances’ operation through which the energy demand issues in residential buildings are predicted and optimized. As a part of the smart home/smart grid, electric vehicles (EV) can be used to balance the demand and supply. Mohammad et al. [32] have proposed an energy management unit for residential buildings with local PV power generation to maximize the user comfort, including the availability of EVs, PAR reduction, and minimize the energy consumption costs.

The authors of [33] present a distributionally robust optimization algorithm to optimally schedule the energy storage system that is integrated with a PV source. This problem has been presented as a two-stage programming model. The first stage reduces the energy consumption costs, and the second stage includes a real-time dispatch with a forecasted PV power output. With system uncertainties such as DC voltage fluctuation, disturbance from the utility grid system, and variation of the circuit parameters, traditional linear control methods cannot ensure the quality issues of the grid-connected inverter system. The authors of [34] propose a robust model predictive control (RMPC) technique that effectively schedules the battery energy storage system to minimize the total economic cost of multicrocar microgrids.

Thus, for effective energy management, this paper proposes a novel optimization algorithm with the Indian electricity pricing scheme to schedule consumers’ demands. For the first time, the Indian DAP scheme was implemented along with the SSA, SSO and proposed RMSSO algorithms to reduce energy consumption cost and PAR. Timing and energy constraints were defined. Additionally, a variable was defined to ensure the user satisfaction level. The system was supported only by grid supply. The best sperm position was determined with the help of a remodeled inertia weight/constriction function. This paper employed qualitative and quantitative metrics to check and validate the correctness and accuracy of the proposed optimization algorithm. The proposed system used an effective communication technology to schedule energy demand in the most economical way.

**Highlights and Organization of the Paper**

The following features make this approach more distinct from existing DR algorithms.

(i). A remodeled sperm swarm optimization (RMSSO) algorithm was proposed for the HEM system.

(ii). The optimization process was carried out with varied computational parameters to demonstrate that the optimization algorithms could handle five distinct Indian climatic conditions.

(iii). A day-ahead pricing (DAP-₹/kWh) scheme was used as a part of the DR program.

(iv). This paper provides a unique comparison of SSO, modified SSO (MSSO), and the proposed RMSSO algorithms.

(v). Reduction in energy consumption costs, peak-average ratio (PAR), and increase in the level of user comfort were the objectives of this paper.
The remaining part of the paper is structured as follows: The proposed system architecture is explained in Section 2, and the mathematical formulation is described in Section 3. Section 4 discusses the proposed RMSSO algorithm. The simulation results, evaluation, and description of the experimental setup are presented in Section 5. Finally, this paper is concluded in Section 6.

2. System Architecture

The proposed system aimed to reduce energy consumption costs by collecting and evaluating all electricity-related data to provide an optimal solution. This system gives a simulation/hardware-based solution for controlling and monitoring the energy in a lab environment. All five scenarios (climates) were developed with a controller device for smooth integration and operation of home appliances. The proposed system comprised 16 appliances with different power ratings, each controlled by an individual relay switch actuated by the controller for every time slot. These appliances were divided into two groups (schedulable and non-schedulable) to simplify operation and improve consumer satisfaction. The setup includes incandescent lamps, a mixer, and a kettle. Table 1 shows the type of loads which are connected across each phase.

Table 1. Load details.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Load Type</th>
<th>Wattage</th>
<th>Total Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>Incandescent</td>
<td>200 W &amp; 100 W</td>
<td>2 each (total 4 nos)</td>
</tr>
<tr>
<td>Y</td>
<td>Incandescent</td>
<td>60 W &amp; 40 W</td>
<td>5 nos and 4 nos, respectively</td>
</tr>
<tr>
<td>B</td>
<td>CFL</td>
<td>9 W</td>
<td>3 nos</td>
</tr>
</tbody>
</table>

The HEMs laboratory setup consisted of a controller (Raspberry Pi 3B+), a smart meter (Schneider Conzerv EM6400NG-model-NHA2768503-0104/2018), RS485 communication modules that employ the MODBUS protocol and loads. The algorithm was developed on the Raspberry Pi, and its input/output options were enabled to connect to the internet through Wi-Fi communication to manage and monitor the appliances. The smart meter (gateway) was connected to the controller via an RS485 module. The IP modem (Four-Faith) and the control unit were Wi-Fi-enabled for remote access. A dynamic domain name system (DDNS) was used when the server IP was dynamic. The modem communicated with the smart meter using the same settings. A Python script read data from smart meter data registers using the pymodbus package [35] and stored it in a local database which was available in the Raspberry Pi.

The timestamp, active power, reactive power, apparent power, frequency, power factor, current, and voltage were recorded. The database got updated every second. Every minute, the controller calculated and recorded the average of all collected fields. If the smart meter failed to read or send the readings to the controller for more than 60 s, the controller sent an alert notification message. Several privacy and security standards are described in [36], and the proposed prototype system uses HTTPS for secure data transmission between local databases and cloud storage. The controller uploaded all collected data as a CSV file to the cloud storage (Thingspeak).

3. Mathematical Formulation

The scheduling process was modeled as a MILP problem that was addressed using problem formulation, decision variables, cost functions, and constraints, and described in the upcoming subsections.

3.1. Problem Formulation

In the scheduling process, sixteen appliances with different power ratings are considered, which are categorized into two groups: schedulable appliances (SA), and non-schedulable appliances (NSA). Table 1 lists the power rating of appliances used in schedul-
ing. The appliance’s operating duration is split into equal intervals of time (1 h each slot) of the day (K) as shown in Equations (1) and (2).

\[ K = k_1, k_2, k_3, \ldots, k_n \]  

(1)

where \( K \) denotes the cumulative number of time intervals in a day (24 h), \( k_n \) represents the \( n \)th time interval, and \( n \) is the number of time intervals per day = \((1, 2, 3, \ldots, N)\) and is as follows,

\[ n = \frac{24 \text{ hour of the day}}{\text{no. of intervals}} = \frac{24 \text{ hour of the interval}}{24 \text{ intervals}} = 1 \text{ hour per interval} \]  

(2)

The group of appliances that are considered for scheduling is denoted as \( G \), and this is a combination of both schedulable and non-schedulable appliances, as shown in Equation (3).

\[ G = g_1, g_2, g_3, \ldots, g_n \]  

(3)

where \( g_1, g_2, g_3, \ldots, g_n \) specifies the individual appliance.

3.2. Binary Decision Variable

The binary variable \( \left(b_{in}^{k_n}_{g_n,j}\right) \) is formulated to determine whether the considered set of schedulable and non-schedulable appliances is in an ON or OFF condition, as shown in Equation (4).

\[ b_{in}^{k_n}_{g_n,j} \in 0, 1 \]  

(4)

If \( b_{in}^{k_n}_{g_n,j} = 1 \) for appliance \( g_n \), the \( j \)th set has been scheduled in the time interval \( k_n \).

An additional binary decision variable \( \left(b_{in}^{k_n}_{g_n,j}\right) \) is defined to determine whether a specific \( j \)th set of appliances has finished their operation by the time \( k_n \), and the same is given in Equation (5).

\[ b_{in}^{k_n}_{g_n,j} = 1 \]  

(5)

As discussed in earlier sections, the aim is to reduce the total electricity usage costs. The total energy consumption costs are computed using Equation (6) with a 24 h electricity tariff: day-ahead price (DAP- (₹/kWh)) [37].

\[ \sum_{k=1}^{24} \pi_{kn}^{\text{price}} \left( \sum_{g=1}^{n} \sum_{j=1}^{g} p_{ON}^{Total}\right) \]  

(6)

where \( \pi_{kn}^{\text{price}} \) represents the electricity tariff for the respective time slots \( (k_n) \), and the total energy consumed by all appliances \( (p_{ON}^{Total}) \) on the particular day is determined using Equation (7).

\[ p_{ON}^{Total} = p_{NSA}^{ON} + p_{SA}^{ON} \]  

(7)

Equations (8) and (9) describe the amount of energy consumed by schedulable and non-schedulable appliances scheduled in the appropriate time interval \( (k_n) \).

\[ p_{NSA}^{ON} = \sum_{k=1}^{24} \left( \sum_{g=1}^{G_{NSA}} p_{NSA}^{g_k_n} \right) = \left[ p_{NSA}^{g_1k_1} + p_{NSA}^{g_2k_2} + \ldots + p_{NSA}^{g_{24}k_{24}} \right] \]  

(8)

\[ p_{SA}^{ON} = \sum_{k=1}^{24} \left( \sum_{g=1}^{G_{SA}} p_{SA}^{g_k_n} \right) = \left[ p_{SA}^{g_1k_1} + p_{SA}^{g_2k_2} + \ldots + p_{SA}^{g_{24}k_{24}} \right] \]  

(9)
where $G_{SA}$ and $G_{NSA}$ are the sets of SA and NSA appliances, $P_{NSA^{ON}}$ and $P_{SA^{ON}}$ are the power consumed by non-schedulable and schedulable appliances, respectively. $P_{ON^{Total}}$ is the total power consumption of all appliances.

3.3. Constraints

Two types of constraints are described in this section: timing and energy constraints.

3.3.1. Timing Constraints

1. Non-schedulable appliance: As specified in Equation (10), non-schedulable appliances must remain ON throughout the day (24 h), regardless of whether it is a peak time or not.

$$\sum_{k=1}^{24} K = \left| G_{NSA^{ON}} \right|$$  \hspace{1cm} (10)

2. User satisfaction level: The significant constraint is that all appliances must be run for a specified number of times $(T_{app,x})$, as defined in Equation (11).

$$\sum_{k=1}^{24} T_{app,x} = t_{appN}$$  \hspace{1cm} (11)

where $T_{app,x}$ refers to the total number of times that an appliance has to be operated. The appliance type (SA and NSA) is indicated as $appN$. An appliance is operated for the desired number of times per day. The frequency of appliance operation (provided by the consumer) is referred to as $appx$, which guarantees that all appliances are operated the required number of times, as shown in Equation (12).

$$\sum_{k=1}^{n} t_{app,x} + t_{app,x+1} + t_{app,x+2} + \ldots + t_{app,x+(N-1)} = T_{app,x}$$  \hspace{1cm} (12)

In addition, constant for the availability of appliances, three schedulable appliances, such as $(A02, A05, A08)$, are considered unavailable during the time slots $k_1, k_2, k_3$, respectively, to reduce peak demand issues, as given in Equation (13).

$$x(appN,k_n) \in \{0\}$$  \hspace{1cm} (13)

3. Time constraint: The appliance operating time constraint specifies the scheduled time interval of the $j$th set of appliances as given in Equation (14).

$$bin_{g_n,j}^{k_n} \leq K_{g_n}^{k_n} \hspace{1cm} \forall g_n, j, k_n$$  \hspace{1cm} (14)

where $K_{g_n}^{k_n}$ is the scheduled time interval of the $j$th set of appliances.

3.3.2. Energy Constraints

1. Energy consumption threshold limit $(E_{max})$: For any time interval of the day, the total power consumed by both schedulable and non-schedulable appliances $(P_{ON^{Total}})$ must be less than or equal to the threshold limit, $E_{max} = 1.2$ kW, as stated in Equation (15).

$$P_{ON^{Total}} \leq E_{max}$$  \hspace{1cm} (15)

After the scheduling process, if the condition $P_{ON^{Total}} < E_{max}$ still exists, consumers can turn ON additional appliances (part of schedulable loads which are not involved in...
the scheduling for the specific time interval). This condition takes care of the consumers’ satisfaction level. Then, the total power consumption is as follows:

\[ P_{ON}^{\text{Total}} = P_{NSA}^{ON} + P_{SA}^{ON} + P_{\text{additional}}^{ON} \]  

(16)

where \( P_{\text{additional}}^{ON} \) is the additional appliances’ power consumption. If \( P_{ON}^{\text{Total}} > E_{\text{max}} \) or \( P_{ON}^{\text{Total}} > E_{\text{max}} \), the additional appliance that the user desires to operate is not feasible to run.

In continuation of this, NSA minimal demand \( (E_{\text{min}}) \) is stated in Equation (17).

\[ E_{\text{min}} \leq P_{ON}^{\text{Total}} \leq E_{\text{max}} \]  

(17)

where \( E_{\text{min}} \) and \( E_{\text{max}} \) represents the lower and upper power limitations to the \( j \)th set of appliances, respectively.

2. Total energy consumption: This constraint (Equation (18)) is imposed to guarantee that, through scheduling, the required overall energy demand \( (E_{\text{ON},i} = 10 \text{ kW}) \) of the day has been met.

\[ \sum_{k=1}^{24} P_{ON}^{\text{Total}} = E_{\text{ON},i} \quad \forall g_{\text{ON}}, j, k \]  

(18)

3. Peak-average ratio (PAR): The peak-average ratio is an important factor to consider while scheduling and must be mitigated. It is computed using Equation (20).

\[ P_{\text{peak}} = \max \left( \sum_{t=1}^{24} (P_{ON}(t)) \right) \quad \text{and} \quad P_{\text{avg}} = \frac{P_{ON}^{\text{Total}}}{D} \]  

(19)

\[ \text{PAR} = \frac{P_{\text{peak}}}{P_{\text{avg}}} \]  

(20)

where \( P_{\text{peak}} \) and \( P_{\text{avg}} \) are the maximum power and average power of the day, respectively.

3.3.3. Objective Function

After meeting the above-stated constraints, the optimal energy consumption cost can be attained and consequently, the objective function \( (OF) \) of this paper is as defined in Equation (21).

\[ OF = \min \left( \sum_{k=1}^{24} \pi_{\text{price}}^{\text{ON}} \sum_{g_{\text{ON}}} \sum_{j} \sum_{k} \sum_{n_{\text{ON}}}^{P_{\text{ON}}^{\text{Total}}} \right) \]  

(21)

where \( P_{\text{ON}}^{\text{Total}} \) is the total power consumption of all scheduled appliances in the specific time interval. \( \pi_{\text{price}}^{\text{ON}} \) indicates the cost of energy consumption. As a consequence, Equation (21) is the optimization model of MILP. It can be efficiently solved using metaheuristic methods such as SSO, SSA, and the proposed RMSSO algorithms.

4. Optimization Techniques

A novel remodeled sperm swarm optimization (RMSSO) algorithm was proposed for home energy optimization. For the first time, an Indian home environment was taken as the case study to solve this optimization problem using the salp swarm optimization (SSA) and sperm swarm optimization (SSO) algorithms. All three algorithms were developed using the Python/GUROBI tool without integrating renewable energy sources.

4.1. Salp Swarm Optimization Algorithm (SSA)

The salp swarm optimization algorithm (SSA) mimics the salps group (family of Salpidae) foraging mechanism in the oceans. Salps are intelligent and have translucent
barrel-shaped bodies, similar to those of jellyfish. A salp chain is a network (swarm) formed by a group of salps in the deep sea [38].

The formation of swarms (salp chain) evolves into a hierarchy with a leader and followers. The first salp in the chain is the leader, while the rest of the salps are followers. The leader salp communicates either directly or indirectly with the follower salps using search directions. The positions of all salps are defined as \( X_j^d = [x_j^d, y_j^d, \ldots, x_{n_j}^d, y_{n_j}^d] \), where \( d = 1, 2, 3, \ldots; n \) is the number of salps; and \( j = 1, 2, \ldots, m; m \) is the number of variables. The SSA assumes that the food source \((F)\) is available in the search area as a target. The position of leader salp is defined as in Equation (22), with three components: direction, personal best, and the team’s best.

\[
X_j^1 \begin{cases} 
F_j + \lambda_1 ((u_j - l_j)\lambda_2 + l_j), & \lambda_3 \geq 0 \\
F_j - \lambda_1 ((u_j - l_j)\lambda_2 + l_j), & \lambda_3 < 0
\end{cases}
\]  

(22)

where \( X_j^1 \) denotes the leader salp’s position at the \( j \)th dimension in the search space, \( F_j \) is the position of food at the \( j \)th dimension in the search space, \( \lambda_1, \lambda_2, \lambda_3 \) are the coefficient factors, \( u_j \) and \( l_j \) indicate the upper and lower bounds of the \( j \)th dimension. The significant controlling coefficient parameter of this algorithm is \( \lambda_1 \). During the optimization process, this parameter decreases as the iteration count increases, allowing the SSA technique to explore more in the early phases and then exploit it extensively in the search space. The coefficient factors \( \lambda_2 \) and \( \lambda_3 \) are consistently generated random numbers, where \( \lambda_2 \) is in the range of \([0, 1]\), which is responsible for widening the search space. \( \lambda_3 \) indicates whether the next positions of the current leader salp and follower salps are within the boundary or not. If \( \lambda_3 < 0.5 \), the salps are moving out of the boundary on a negative scale, while if \( \lambda_3 \geq 0.5 \), salps are travelling towards the direction of food on a positive scale. The boundary for \( \lambda_3 \) is fixed as 0.5 to ensure equal weightage is given to the salps travelling in both forward and reverse directions. Thus, \( \lambda_2 \) and \( \lambda_3 \) help decide the next position of the salp in the \( j \)th dimension of the search space. Additionally, the coefficient factors \( \lambda_1, \lambda_2, \) and \( \lambda_3 \) are used to reposition the solution that goes outside the search space. The position of salp followers is updated using Equation (23).

\[
x_j^d = \frac{1}{2} (X_j^d + X_j^{d-1})
\]

(23)

where \( x_j^d \) represents the position of the \( d \)th follower in the \( j \)th dimension in the search space. Equation (24) shows how to bring the salp back into the search space.

\[
X_j^d = \begin{cases} 
l_j, & \text{if } X_j^d \leq l_j \\
u_j, & \text{if } X_j^d \geq u_j \\
X_j^d, & \text{otherwise}
\end{cases}
\]

(24)

The SSA optimization is started by initializing the salps in a random position. Consequently, the fitness of every single salp is determined based on the distance between the food source \((F)\) and the salp. For each dimension, with the help of coefficient factors \((\lambda_1, \lambda_2, \) and \( \lambda_3)\), the positions of both leader and follower salps are updated frequently. \( X_j \) is considered as the optimum load scheduling for the cost-saving of a day (24 h). The salp chain exploits the search space to get the most appropriate global optimum solution and avoids the local optimum.

4.2. Sperm Swarm Optimization Algorithm

This section discusses the sperm swarm optimization (SSO) algorithm. The SSO mimics the sperm navigation and mobility to inseminate the ovum. The swarm swims from the cervix, a low-temperature region, to the fallopian tubes, a high-temperature region, where the egg waits to be fertilized by the swarm. Sperm migration has to meet
constraints, like pH value and temperature inside the reproductive system. Finally, one sperm penetrates and fertilizes the egg, which is referred to as the winner [39]. The symmetrical side stroke of the sperm swarm is shown in Figure 1.

![Symmetrical side stroke of sperm swarm and the egg.](image)

**Figure 1.** Symmetrical side strokes of sperm swarm and the egg.

The sperm swarm optimization algorithm reaches an optimal solution by updating the individual’s current position. This is achieved by determining the initial velocity of sperm \( (Initial_{Velocity}) \), the distance between the current position and \( sbsolution \) (the sperm’s best solution achieved so far), and \( sgb solution \) (the sperm’s global best position). The mathematical model of SSO can be represented as,

\[
V_i(t) = Initial_{Velocity} + Current_{Bestposition} + Global_{Bestposition}
\]

where \( V_i(t) \) is the velocity of sperm \( i \) \( (i = 1, 2, 3, \ldots, N) \); \( N \) is the maximum sperm counts at iteration \( t \). The initial velocity, personal best position, and global best position of the sperm are determined using Equations (26)–(29).

\[
Initial_{Velocity} = D \cdot V \cdot \log_{10}(pHRand1)
\]

\[
Current_{Bestposition} = \log_{10}(pHRand2) \cdot \log_{10}(TempRand1) \cdot (sbsolution - current_i)
\]

\[
Global_{Bestposition} = \log_{10}(pHRand3) \cdot \log_{10}(TempRand2) \cdot (sgb solution - current_i)
\]

\[
current_i = current_i + V_i
\]

where \( D \) is the velocity (inertia weight) damping parameter, a random number between 1 and 0 that is used to control and regulate the sperm velocity. \( pHRand1 \), \( pHRand2 \), and \( pHRand3 \) are the random numbers of the pH value that varies between 7 and 14. The \( TempRand1 \) and \( TempRand2 \) are the random temperatures that range between 35.1 and 38.5 °C.

SSO uses an inertia weight that converges to an optimal value over the course of iteration which is represented as \( D \) in Equation (26). There are two linear methods of determining the inertia (constriction) weight which are given in Equations (30) and (31).

\[
D_{i+1} = D_{max} - i \times \frac{D_{max} - D_{min}}{i_{max}}
\]

\[
D_{i+1} = \Delta D \cdot D_i
\]

where \( i_{max} \) is the maximum iteration, \( i \) is the current iteration, \( D_{max} \) and \( D_{min} \) are the upper and lower limits of the velocity damping parameter. \( \Delta D \) denotes the random value that varies between 1 and 0. The second linear (dynamic) method outperforms the first method in terms of convergence rate.

### 4.3. Remodeled Sperm Swarm Optimization Algorithm (RMSSO)

**Constriction Coefficient of the Sperm Swarm Optimization Algorithm**

As discussed in Section 4.2, the authors of [39] have presented the sperm swarm optimization using the linear Equation (31) that results in sperm convergence over the course of iterations. That is, the magnitude of sperm decreases linearly (adaptively) as it focuses more on the local and previous best points [40]. For low-dimensional optimization problems, this algorithm converges to the optimal point over a period of time. However, when dealing with high-dimensional and complicated energy optimization problems, this SSO is not capable of giving a promising solution [41]. Therefore, to improve the global
search capability of the SSO algorithm, the constriction coefficient of SSO was restructured and proposed as a remodeled SSO algorithm in this paper.

For the past 350 years, researchers believed that sperm travels to the egg by swimming, as discussed in Section 4.2. Gadêlha et al. [42] has found that the sperm actually swims by spinning in a helical shape, as shown in Figure 2. Researchers have developed various optimization algorithms using different spiral trajectories such as Archimedes spiral, Cycloid spiral, Epitrochoid spiral, Hypotrochoid spiral, Logarithmic spiral, Rose spiral, Inverse spiral, and Overshoot spirals [43]. Despite all these trajectories being fast in computation, due to inadequate exploration of the search space, this technique eventually converges to the local optimum value [44,45]. It is worth mentioning here that the most commonly used spiral trajectory is the logarithmic spiral. The logarithmic spiral is also referred to as an equiangular or growth spiral, owing to the fact that the spiral distance increases with the number of iterations. In metaheuristic techniques, the unique processes of engendering a logarithmic spiral have been realized as an effective search behavior [46–48].

Figure 2. The sperm swarm helical spin in the fallopian tubes.

In the proposed remodeled sperm swarm optimization algorithm, the constriction mechanism allows the algorithm to find the best solution by continuously avoiding the trapping of local solutions. The mathematical model of sperm’s helical shape movement is given in Equation (32). This equation represents the nature of helical movement of the sperm swarm. The radius (D-inertia weight) of the sperm movement is decreased as the iteration count increases.

\[ D_{i+1} = 2e^{-\left(\frac{4m}{M}\right)^2} \]  

(32)

where \( m \) and \( M \) are the current and maximum number of iterations, respectively. Thus, the inertia parameter is adaptively decreased from 1 to 0. Figure 3 illustrates the RMSSO algorithm’s search process from diversification to intensification, and the flow chart of the proposed RMSSO algorithm is illustrated in Figure 4.

Figure 3. Illustration of the RMSSO algorithm’s sperm swarm and the winner process.
Figure 4. Flowchart of the proposed RMSSO algorithm.

4.4. Comparison Optimization Algorithms Inspired by the Sperm Swarm

A unique comparison of optimization algorithms inspired by the sperm swarm is made and illustrated in Table 2.

Table 2. Comparison of optimization algorithms inspired by the sperm swarm.

<table>
<thead>
<tr>
<th>Comparison Standards</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sperm Swarm Optimization (SSO) [39], 2018</td>
</tr>
<tr>
<td></td>
<td>Modified Sperm Swarm Optimization (MSSO) [41], 2021</td>
</tr>
<tr>
<td></td>
<td>Remodeled Sperm Swarm Optimization (RMSSO) Proposed</td>
</tr>
<tr>
<td>Metaphor type</td>
<td>Nature-inspired approach, mimics the motility of sperm swarms during the fertilization process.</td>
</tr>
<tr>
<td>Flow of sperm and control parameter</td>
<td>Swim stroke and velocity damping coefficient function.</td>
</tr>
<tr>
<td>Type of approach</td>
<td>It continuously updates the swarm’s position and velocity.</td>
</tr>
<tr>
<td>Fitness value</td>
<td>Use the optimal value of the winner sperm as a reference value to adjust the velocities of the remaining sperms in the swarm.</td>
</tr>
<tr>
<td>Impact of sperm swarms on the solution</td>
<td>Linear</td>
</tr>
<tr>
<td>Results</td>
<td>Local Optimum</td>
</tr>
</tbody>
</table>

5. Results and Discussion

An efficient swarm-based optimization technique balances both local and global search [49]. A suitable balance between these two processes can approximate an optimized
home energy management system. If algorithms pay more attention to local search, a solution will quickly converge to an optimum point and get trapped at a local optimum.

A larger search coefficient (constriction coefficient) on the global search helps to avoid the local optimum solutions, but it takes more computation time to achieve the global optimum solution. The SSA technique uses the first salp (leader) in the swarm chain to balance the local and global search; the global search coefficient (constriction coefficient) decreases with an increase in the number of iterations. Hence, the SSA faces difficulty in achieving a proper balance between local search and global search [30]. In the SSO algorithm, each sperm optimizes its position by considering its location, velocity, the local best solution, and global best solution. However, this algorithm cannot converge at a global minimum and is trapped in local optima and faces premature convergence in complex problems [41]. Thus, in terms of exploration and exploitation, both SSA and SSO algorithms have enough exploration ability, but their exploitation ability is comparatively low.

Therefore, in this paper, a remodeled SSO (RMSSO) algorithm is proposed with an effective modification (Equation (32)) that improves the diversity of solutions by each sperm and keeps the proper balance between local search and global search during optimization. Thus, the searching ability of RMSSO was better than that of SSA and SSO for all five climates. This is the reason why the proposed RMSSO outperformed SSA and SSO consistently in all conditions.

5.1. Simulation Results

The simulation results achieved using SSA, SSO, and the proposed RMSSO algorithms under the day-ahead pricing scheme are compared in this section. A simulation was carried out for sixteen different appliances with a total demand of 10 kW under five climatic conditions and constraints using the Python/GUROBI tool. The specifications of the system used were as follows: processor Intel (R) Core (TM) i3-7020 U CPU @ 2.30 GHz; 12.0 GB RAM; 64-bit operating system type; x64-based processor. All algorithms scheduled the load demand as shown in Figure 5a–e, which are discussed in the upcoming sections. Table 3 illustrates the simulation parameters.

<table>
<thead>
<tr>
<th>Table 3. Simulation parameters of SSA, SSO, and RMSSO techniques.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>pH1, pH2, pH3</td>
</tr>
<tr>
<td>Temp1, Temp2</td>
</tr>
<tr>
<td>Iterations</td>
</tr>
<tr>
<td>$E_{\text{max}}$</td>
</tr>
<tr>
<td>$E_{\text{gnd}}$</td>
</tr>
<tr>
<td>$D_{\text{Max}}$</td>
</tr>
<tr>
<td>$D_{\text{Min}}$</td>
</tr>
<tr>
<td>$\text{dim}$</td>
</tr>
<tr>
<td>$\lambda_2$ and $\lambda_3$</td>
</tr>
</tbody>
</table>

DAP hourly price for five climatic conditions was taken from the Indian Energy Exchange, and the same is represented in Table 4 [37]. Recent research [50–54] has proved that both TOU and DAP pricing schemes reduce the energy consumption cost and the peak-average ratio. Hence, this paper considered the DAP scheme as the preferred option to use. The highlighted price in Table 4 represents the peak price of the day.
4. Results and Discussion

An efficient swarm-based optimization technique balances both local and global optimization processes 
and is trapped in local optima and faces premature convergence in computation time [48]. This is the reason why the proposed RMSSO outperformed SSA and SSO algorithms. A suitable balance between these two processes can approximate an optimum point and get trapped at a local optimum. This is why the proposed RMSSO outperformed SSA and SSO algorithms in all conditions.

Table 4. Day-ahead price—Indian scenario.

<table>
<thead>
<tr>
<th>Hours</th>
<th>Monsoon</th>
<th>Autumn</th>
<th>Spring</th>
<th>Summer</th>
<th>Winter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0–1</td>
<td>2.51985</td>
<td>2.18985</td>
<td>3.71946</td>
<td>4.90168</td>
<td>2.02185</td>
</tr>
<tr>
<td>1–2</td>
<td>2.25144</td>
<td>2.42961</td>
<td>3.36554</td>
<td>4.00521</td>
<td>1.92394</td>
</tr>
<tr>
<td>2–3</td>
<td>2.13724</td>
<td>2.32931</td>
<td>3.28878</td>
<td>3.44357</td>
<td>1.80898</td>
</tr>
<tr>
<td>3–4</td>
<td>2.01204</td>
<td>2.32506</td>
<td>3.23199</td>
<td>3.16281</td>
<td>1.78339</td>
</tr>
<tr>
<td>4–5</td>
<td>2.0372</td>
<td>2.38732</td>
<td>3.29752</td>
<td>3.40298</td>
<td>1.9668</td>
</tr>
<tr>
<td>5–6</td>
<td>2.1493</td>
<td>2.43323</td>
<td>3.91974</td>
<td>3.44873</td>
<td>2.15122</td>
</tr>
<tr>
<td>6–7</td>
<td>2.55065</td>
<td>2.50596</td>
<td>4.87516</td>
<td>3.24231</td>
<td>2.81923</td>
</tr>
<tr>
<td>7–8</td>
<td>2.52455</td>
<td>2.43629</td>
<td>5.30393</td>
<td>2.78748</td>
<td>4.27534</td>
</tr>
</tbody>
</table>

Figure 5. Demand comparison of SSA, SSO, and RMSSO techniques.
In the monsoon, spring, and summer seasons, one non-schedulable appliance is taken into account, and in the autumn and winter seasons, two non-schedulable appliances were considered for 24 h operation. Table 5 represents the number of times that an appliance is to be operated to ensure that a specific appliance task is completed, and the same is achieved 100% by satisfying the constraints provided in Equations (12)–(19). The highlighted numbers in Table 5 represent the number of times the appliance (non-schedulable) must be operated.

### Table 5. Number of times that an appliance has to be operated.

<table>
<thead>
<tr>
<th>Appliances</th>
<th>Monsoon</th>
<th>Autumn</th>
<th>Spring</th>
<th>Summer</th>
<th>Winter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A01</td>
<td>21</td>
<td>3</td>
<td>5</td>
<td>24</td>
<td>12</td>
</tr>
<tr>
<td>A02</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>23</td>
<td>20</td>
</tr>
<tr>
<td>A03</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>A04</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>A05</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>A06</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>21</td>
<td>7</td>
</tr>
<tr>
<td>A07</td>
<td>24</td>
<td>21</td>
<td>19</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>A08</td>
<td>23</td>
<td>6</td>
<td>23</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>A09</td>
<td>5</td>
<td>4</td>
<td>21</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>A10</td>
<td>18</td>
<td>24</td>
<td>24</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>A11</td>
<td>9</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>A12</td>
<td>5</td>
<td>19</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>A13</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>A14</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>A15</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>A16</td>
<td>5</td>
<td>24</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
5.1.1. Demand Comparison

Figure 5a–e shows the comparison of load scheduling using the SSO, SSA, and proposed RMSSO techniques for individual time slots. The total load request by the consumer during the day was limited to the threshold of 10 kW to avoid peak demand issues. In Figure 5a, (monsoon) the $t_{20}$, $t_{21}$, and $t_{22}$ time slots with high electricity prices show that both SSO and SSA algorithms had scheduled the load of 0.4 to 0.6 kW, whereas RMSSO scheduled 0.2 to 0.3 kW, which is less than the load scheduled by SSO and SSA. Similarly, during the $t_{23}$ slot, SSO and SSA scheduled 0.3 kW of demand. However, RMSSO scheduled the appliances that had a total load of 0.2 kW. Note that the proposed RMSSO distributed the appliance across all time intervals, resulting in a reduced energy consumption cost without exceeding the threshold limit of 1.2 kW.

Likewise, in the remaining climatic conditions (autumn, spring, summer, and winter), the proposed RMSSO effectively scheduled load in all time slots in comparison to other techniques, as illustrated in Figure 5b–e. Consequently, this comparison reveals that the proposed RMSSO algorithm scheduled the appliance with a reduced cost of energy consumption and regulated peak demand, which is a benefit for both the utility and the consumer.

5.1.2. Total Load Comparison

Figure 6 presents the total load comparison results of all three algorithms. The proposed RMSSO scheduled its load at 10 kW in all five climatic conditions. Considering the autumn season, both SSO and RMSSO scheduled the loads with 9.43 and 10 kW respectively, whereas SSA scheduled the load with 14.5 kW. In the summer season, both SSA and SSO scheduled the load of 16.43 kW each, which was greater than the total load requirement (10 kW) for the day. This excess load scheduling causes concerns such as power loss, increased peak demand, and the high cost of energy consumption. The RMSSO algorithm covered the total demand of 10 kW, which indicated that 100% of the consumer’s maximum demand was satisfied.

![Figure 6. Comparison of total scheduled loads.](image)

5.1.3. Cost Comparison

Figure 7a–e shows the cost comparison of RMSSO with those of SSA and SSO techniques for each time slot. Considering Figure 7a with day-ahead pricing (monsoon), the RMSSO algorithm achieved energy consumption costs lower than those of SSO and SSA between 0.90₹ and 1.15₹ during the high-cost time ($t_{20}$, $t_{21}$, and $t_{22}$) slots. At the same time, the SSA and SSO techniques scheduled the load between 1₹ and 2.25₹. In comparison, the RMSSO energy consumption costs were lesser than those of the SSA and SSO algorithms in all the time slots. Figure 7b shows the individual time slot cost comparison in the autumn season. Notably, in the peak period ($t_{19}$ to $t_{23}$), the SSA showed maximum energy consumption cost, and SSO and RMSSO showed around 0.5₹ to 1₹. RMSSO scheduled with the energy consumption cost of 2.5₹ in the $t_{5}$ time slot, which was higher than the load scheduled by the SSA technique, indicating that RMSSO scheduled higher demand than the other techniques did in these slots. Nonetheless, even in the other climates (Figure 7c–e),
the proposed RMSSO algorithm outperformed the other two algorithms in saving the energy consumption cost.

![Graphs showing energy consumption cost comparison](image)

**Figure 7.** Individual time slot cost comparison of SSA, SSO, and RMSSO techniques.

### 5.1.4 Total Cost Comparison

The total cost comparison of the proposed RMSSO with those of SSA and SSO is given in Figure 8. Considering the monsoon season, the RMSSO energy consumption cost was 24.11 INR, which was less than those of the SSA and SSO techniques, i.e., 30.88 INR and 31.91 INR, respectively. Similarly, during the autumn season, SSO and RMSSO attained the lowest costs of 24.30 INR and 24.15 INR, respectively, compared to that of the SSA technique. Accordingly, the proposed RMSSO stood ahead of SSO and SSA techniques. Table 6 provides a detailed comparison of the total energy consumption cost and load scheduled by each technique.
The optimization process was carried out with varied climatic conditions. The controller uploaded all collected data as a CSV file to the cloud storage (Thingspeak). The controller was actuated using an HTTPS protocol. The controller setup files were sent using an IP modem or a Raspberry controller. The controller was connected to the NHA2768503 smart load (Conzerv Smart Load) using a MODBUS as a communication protocol. The controller disabled all non-schedulable loads and enabled the schedulable loads, whereas RMSSO attained the global optimal solution. Similarly, the comparison of remaining climatic conditions also proved that 100% of the tasks were completed at the required time. The percentage of cost difference between SSO and SSA from RMSSO for the same autumn season was 3.908% and 10.188%, respectively. In the same scenario, the percentage cost difference of SSO and SSA approaches met 164.4% and 164.3% of the requirement, respectively. The proposed RMSSO completely satisfied the demand (100%) for the autumn season, and only 94.3% of the demand was satisfied by the SSO algorithm. This shows that SSO could not meet the total demand requirement. In the same scenario, the SSA technique finished 145.2% of the task. The percentage of cost difference between SSO and SSA from RMSSO for the same autumn season was 0.617% and 37.30%, respectively. Further, while considering the spring season, RMSSO satisfied 100% of the demand required. However, the SSO and SSA techniques were completed at 102.9% and 106.9%, respectively. Additionally, the percentage of cost difference between SSO and SSA from RMSSO in the same season was 3.908% and 10.188%, respectively.

During the summer season, RMSSO met 100% of the required demand, whereas the SSO and SSA approaches met 164.4% and 164.3% of the requirement, respectively. The percentage cost difference of SSO and SSA from that of RMSSO in the summer season was 40.75% and 40.74% respectively. Similarly, RMSSO completed the required demand of 100% in the winter season, while the SSO and SSA techniques accomplished demand of about 150.8% and 144.9% respectively. While considering the percentage cost difference of SSO and SSA techniques from the proposed RMSSO algorithm, both SSO and SSA techniques scheduled their demand with high cost with a difference of 36.48% and 33.53% respectively.

5.1.5. Task Completion Analysis

From Table 7, it is found that in the monsoon season, the proposed RMSSO algorithm completed 100% of the task (10 kW), while the SSO and SSA techniques completed 129.5% and 126.5%, respectively. In the same scenario, the percentage cost difference of SSO and SSA from RMSSO was 24.44% and 21.92%, respectively. The proposed RMSSO completely satisfied the demand (100%) for the autumn season, and only 94.3% of the demand was satisfied by the SSO algorithm. This shows that SSO could not meet the total demand requirement. In the same scenario, the SSA technique finished 145.2% of the task. The percentage of cost difference between SSO and SSA from RMSSO for the same autumn season was 3.908% and 10.188%, respectively. In the same scenario, the percentage cost difference of SSO and SSA approaches met 164.4% and 164.3% of the requirement, respectively. Similarly, RMSSO completed the required demand of 100% in the winter season, while the SSO and SSA techniques accomplished demand of about 150.8% and 144.9% respectively. While considering the percentage cost difference of SSO and SSA techniques from the proposed RMSSO algorithm, both SSO and SSA techniques scheduled their demand with high cost with a difference of 36.48% and 33.53% respectively.

Table 6. Cost for energy consumption and load scheduled.

<table>
<thead>
<tr>
<th>Seasons</th>
<th>RMSSO</th>
<th>SSO</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monsoon</td>
<td>24.11</td>
<td>31.91</td>
<td>30.88</td>
</tr>
<tr>
<td>Autumn</td>
<td>24.15</td>
<td>24.3</td>
<td>38.52</td>
</tr>
<tr>
<td>Spring</td>
<td>38.11</td>
<td>39.66</td>
<td>42.43</td>
</tr>
<tr>
<td>Summer</td>
<td>31.01</td>
<td>52.34</td>
<td>52.33</td>
</tr>
<tr>
<td>Winter</td>
<td>29.87</td>
<td>47.03</td>
<td>44.94</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Seasons</th>
<th>Load Scheduled (kW)</th>
<th>RMSSO</th>
<th>SSO</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monsoon</td>
<td>10</td>
<td>12.95</td>
<td>12.65</td>
<td></td>
</tr>
<tr>
<td>Autumn</td>
<td>10</td>
<td>9.43</td>
<td>14.52</td>
<td></td>
</tr>
<tr>
<td>Spring</td>
<td>10</td>
<td>10.29</td>
<td>10.89</td>
<td></td>
</tr>
<tr>
<td>Summer</td>
<td>10</td>
<td>16.43</td>
<td>16.43</td>
<td></td>
</tr>
<tr>
<td>Winter</td>
<td>10</td>
<td>15.08</td>
<td>14.49</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8. Total cost comparison.
respectively. It was because the SSO and SSA techniques were trapped in local optima solutions, whereas RMSSO attained the global optimal solution. Similarly, the comparison of remaining climatic conditions also proved that 100% of the tasks were completed at the lowest response time with maximum consumer satisfaction by the proposed RMSSO algorithm. Meanwhile, both SSO and SSA techniques scheduled the loads at higher energy consumption costs. Load scheduling by SSO and SSA techniques led to power loss, increased peak demand, and a high cost of energy consumption.

### Table 7. Percentage cost difference and task completion comparison.

<table>
<thead>
<tr>
<th>Seasons</th>
<th>Techniques</th>
<th>RMSSO</th>
<th>SSO</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monsoon</td>
<td>-</td>
<td>24.44</td>
<td>21.92</td>
<td></td>
</tr>
<tr>
<td>Autumn</td>
<td>-</td>
<td>0.617</td>
<td>37.30</td>
<td></td>
</tr>
<tr>
<td>Spring</td>
<td>-</td>
<td>3.908</td>
<td>10.18</td>
<td></td>
</tr>
<tr>
<td>Summer</td>
<td>-</td>
<td>40.75</td>
<td>40.74</td>
<td></td>
</tr>
<tr>
<td>Winter</td>
<td>-</td>
<td>36.48</td>
<td>33.53</td>
<td></td>
</tr>
</tbody>
</table>

### Table 8. Best, average, and worst outcomes obtained from the proposed RMSSO, SSO, and SSA techniques.

<table>
<thead>
<tr>
<th>Seasons</th>
<th>Program Run Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monsoon</td>
<td>0.42</td>
</tr>
<tr>
<td>Autumn</td>
<td>0.57</td>
</tr>
<tr>
<td>Winter</td>
<td>0.59</td>
</tr>
</tbody>
</table>

5.1.6. Robustness

An effective optimization algorithm should converge to the same global solution over iterations. The convergence curve of the proposed RMSSO algorithm was compared with those of the SSO and SSA techniques (Figure 9a–e). These convergence curves were plotted against the number of iterations (50 counts). From the figure, it is proven that the RMSSO algorithm attained energy consumption costs lower than those with the other algorithms in all five climate conditions, which was because of the effective modification of Equation (32). Further, the curves prove that RMSSO was capable of exhaustively exploring and exploiting the search space in order to determine the best optimal cost with the lowest response time.

This paper adopted and used existing methodologies to check and validate the correctness and accuracy of the proposed optimization algorithm. Firstly, the quantitative results were used to measure how much better RMSSO was compared to SSA and SSO algorithms. Exploration occurs before exploitation, which supports RMSSO to improve the accuracy towards global optimum. To provide a fair comparison, the controlling parameters for all the algorithms, like number of search agents and maximum iteration, were kept same (i.e., 50). The dimension considered was also common to all algorithms (i.e., 24) along with the boundary limit of \( E_{\text{min}}, E_{\text{max}} \), as given in Equation (17). For other controlling parameters, the respective mathematical model of algorithms was used, and its best performance was obtained.

Table 8 illustrates the best, average, and worst outcomes obtained from the proposed RMSSO, SSO, and SSA techniques (which were tuned for 20 trail runs) in the scheduling process. These computational results were obtained from the common home environment, constraints, number of search agents, and number of iterations. This quantitative analysis was done to benchmark the performance of the proposed RMSSO algorithm, which can solve challenging problems even with a large number of variables. Furthermore, from this table, the performance difference between the RMSSO, SSO, and SSA techniques has become more pronounced. While RMSSO attained optimal cost in all five climates, the
SSO and SSA techniques showed poor performance by attaining high energy consumption costs in all climates. Thus, the results have proved that the proposed RMSSO algorithm outperformed the SSA and SSO algorithms.

### Table 7. Percentage cost difference and task completion comparison.

<table>
<thead>
<tr>
<th>Seasons</th>
<th>RMSSO</th>
<th>SSO</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monsoon</td>
<td>24.44%</td>
<td>21.92%</td>
<td></td>
</tr>
<tr>
<td>Autumn</td>
<td>0.617%</td>
<td></td>
<td>37.30%</td>
</tr>
<tr>
<td>Spring</td>
<td>3.908%</td>
<td></td>
<td>10.18%</td>
</tr>
<tr>
<td>Summer</td>
<td>40.75%</td>
<td></td>
<td>40.74%</td>
</tr>
<tr>
<td>Winter</td>
<td>36.48%</td>
<td></td>
<td>33.53%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Seasons</th>
<th>RMSSO</th>
<th>SSO</th>
<th>SSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monsoon</td>
<td>100</td>
<td>129.5</td>
<td>126.5</td>
</tr>
<tr>
<td>Autumn</td>
<td>100</td>
<td>94.3</td>
<td>145.2</td>
</tr>
<tr>
<td>Spring</td>
<td>100</td>
<td>102.9</td>
<td>108.9</td>
</tr>
<tr>
<td>Summer</td>
<td>100</td>
<td>164.4</td>
<td>164.3</td>
</tr>
<tr>
<td>Winter</td>
<td>100</td>
<td>150.8</td>
<td>144.9</td>
</tr>
</tbody>
</table>

| Program Run Time (seconds) | 0.42 | 0.57 | 0.59 |

#### Figure 9. Convergence characteristics’ comparison of RMSSO, SSO, and SSA techniques.
Table 8. Best, average, and worst optimal cost comparison.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Climates</th>
<th>Best (INR)</th>
<th>Average (INR)</th>
<th>Worst (INR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSSO</td>
<td>Monsoon</td>
<td>24.1101</td>
<td>24.1120</td>
<td>24.2144</td>
</tr>
<tr>
<td></td>
<td>Autumn</td>
<td>24.1516</td>
<td>24.1631</td>
<td>24.3376</td>
</tr>
<tr>
<td></td>
<td>Spring</td>
<td>38.1102</td>
<td>38.1127</td>
<td>38.2670</td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td>31.0100</td>
<td>31.0126</td>
<td>31.1692</td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td>29.8708</td>
<td>29.8847</td>
<td>29.8894</td>
</tr>
<tr>
<td>SSA</td>
<td>Monsoon</td>
<td>30.8829</td>
<td>30.8926</td>
<td>30.9196</td>
</tr>
<tr>
<td></td>
<td>Autumn</td>
<td>38.5231</td>
<td>38.5291</td>
<td>38.5329</td>
</tr>
<tr>
<td></td>
<td>Spring</td>
<td>42.4306</td>
<td>42.4660</td>
<td>42.5368</td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td>52.3312</td>
<td>52.3333</td>
<td>52.4259</td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td>44.9415</td>
<td>44.9422</td>
<td>44.9514</td>
</tr>
<tr>
<td>SSO</td>
<td>Monsoon</td>
<td>31.9102</td>
<td>31.9124</td>
<td>31.9262</td>
</tr>
<tr>
<td></td>
<td>Autumn</td>
<td>24.3009</td>
<td>24.3119</td>
<td>24.3991</td>
</tr>
<tr>
<td></td>
<td>Spring</td>
<td>39.6604</td>
<td>39.6761</td>
<td>39.6776</td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td>52.3410</td>
<td>52.3540</td>
<td>52.3544</td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td>47.0321</td>
<td>47.1512</td>
<td>47.2178</td>
</tr>
</tbody>
</table>

5.1.7. Computational Complexity Analysis

In comparison with that of the SSO algorithm, the time complexity (quantitative analysis) of the proposed RMSSO algorithm mainly depends on two aspects: (1) random initialization and (2) sperm velocity and location/position updates. These two aspects qualitatively describe the algorithm’s time complexity and are expressed as $O(N \times D)$ by the Big O notation, where $N$ is the population size and $D$ is the search space dimension. The authors of this paper did not modify the algorithm’s initialization process and the loop body of the algorithm. Therefore, the time complexity was compared in terms of sperm velocity and position updates. Table 9 shows that the proposed RMSSO algorithm had less computational complexity/cost than the traditional sperm swarm optimization (SSO) algorithm and SSA technique did. It was because the proposed RMSSO did not follow the inertia weight updates and the calculation procedure of the SSO algorithm. Despite this, the proposed RMSSO algorithm’s time complexity remained $O(N \times D)$, since the algorithm’s loop body was not altered.

Table 9. Actual computational time.

<table>
<thead>
<tr>
<th>Dimension (D)</th>
<th>Population Size (N)</th>
<th>Computational Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RMSSO</td>
</tr>
<tr>
<td>24</td>
<td>10</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.42</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.58</td>
</tr>
</tbody>
</table>

5.2. Hardware Implementation

A complete experimental model of a home energy management system was designed based on the specification given in Tables 1 and 3, presented in Figure 10a. The power circuit of HEMs is shown in Figure 10b.
Comparison of Peak-Average Ratio (PAR)

In this section, a comparison of the peak-average ratio (PAR) is discussed. From Figure 11, it is observed that a remarkable difference in PAR value was achieved by RMSSO in all five climatic conditions compared to that of SSO and SSA techniques.

Figure 11. Comparison of peak-average ratio.

6. Conclusions

This paper proposed a novel remodeled sperm swarm optimization algorithm for optimal energy usage in an Indian home environment along with multiple constraints to reduce energy consumption cost. For the first time, the SSA, SSO, and RMSSO algorithms were used to solve the energy optimization problem in the Indian scenario. The algorithms were simulated using the Python/GUROBI tool and verified with experimental performance. The SSO, SSA, and proposed RMSSO algorithms were implemented in a common home environment under the DAP (₹/kWh) scheme. The comparison of percentage energy consumption cost difference by SSO and SSA from RMSSO is listed in Table 7 (without RES) for all five climatic conditions. It proved that the proposed RMSSO scheduled the load optimally with 100% task completion (user satisfaction) in less computation time (0.42 s) than the SSO and SSA algorithms did without violating any constraints. The comparative analysis between SSA and SSO showed that both differed by negligible values in their performance and response time. All the comparisons and validation of results were made without incorporating renewable energy sources. Thus, RMSSO could therefore be recommended as an efficient optimization algorithm to be used to control and manage home appliances.

Furthermore, the authors believe that there is a necessity to develop a certain rule-based algorithm that is capable enough to alleviate the intermittent characteristics of the RES with a battery and implement it together in the scheduling process. It is also important
to add more loads so that these algorithms gain significance, which can also be treated as the future scope of this work.

Author Contributions: S.P. R. and P. K. S. have conceived the idea and converted it to an article. The authors confirm their contribution to the paper as follows: P.K.S. encouraged to investigate and supervised the findings of this work, S.P.R. developed the theory and performed the computations, verified the analytical methods, and drafted the article. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by, VELLORE INSTITUTE OF TECHNOLOGY, VELLORE, TAMIL NADU, INDIA. Grant Number: SEED-2634.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used and analyzed during the current study are available from the corresponding author on reasonable request. All the data analyzed during this study is included in this article itself.

Acknowledgments: This research was carried out at Vellore Institute of Technology, Vellore, Tamil Nadu, India. The authors would like to sincerely thank the Management of Vellore Institute of Technology for providing the VIT SEED GRANT to carry out this research work.

Conflicts of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interests.

References


17. Pedrasa, M.A.; Spooner, T.D.; MacGill, I.F. Coordinated scheduling of residential distributed energy resources to optimize smart home energy services. *IEEE Trans. Smart Grid* 2010, 1, 134–143. [CrossRef]


45. Limlawan, V.; Pongchairerks, P. A PSO with ability to avoid being trapped in a local optimum A PSO with ability to avoid being trapped in a local optimum. In Proceedings of the 7th International Joint Conference on Computer Science and Software, Bangkok, Thailand, 12–14 May 2010.


