Quarterly Journal of Optimization In Soft Computing

Vol. 2, Issue 4, Winter 2025

Journal of Optimization of Soft Computing (JOSC)

Vol. 2, Issue 4, pp: (1-15), Winter-2024
Journal homepage: https://sanad.iau.ir/journal/josc

Paper Type (Research paper)

Real-Time Scalable Task Offloading in Edge Computing Using Semi-Markov Decision Processes and Attention-Based Deep Reinforcement Learning

Abbas Mirzaei[1], Naser Mikaeilvand[2], Babak Nouri-Moghaddam[1], Sajjad Jahanbakhsh Gudakahriz[3], Ailin Khosravani[1], Fatemeh Tahmasebizade[1], Ali Seifi[1], Hosein Hatami[1]

1. Department of Computer Engineering, Ardabil Branch, Islamic Azad University, Ardabil, Iran
2. Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran
3. Department of Computer Engineering, Germi Branch, Islamic Azad University, Germi, Iran

Article Info

Article History:
Received: 2024/11/28
Revised: 2025/01/05
Accepted: 2025/02/02

DOI:

Keywords:
Edge Computing; Task Scheduling; Reinforcement Learning; System Scalability.

**Corresponding Author's Email Address: mirzaei_class_87@yahoo.com*

Abstract

Edge computing has emerged as a dynamic framework where computational tasks are offloaded to distributed edge servers (ESs) to provide low-latency and efficient services. As edge systems grow in scale and complexity, leveraging Deep Reinforcement Learning (DRL) has become a prominent approach to optimize task offloading and Resource management. However, traditional DRL-based methodologies encounter several challenges: (1) Discrete-time decision frameworks, such as Markov Decision Processes (MDPs), often enforce offloading in fixed timeslots, leading to scheduling delays and inefficient Resource utilization. (2) Static computational structures struggle to adapt to varying numbers of edge servers or user devices, resulting in scalability issues and system inefficiencies. To overcome these limitations, we introduce a novel DRL-driven real-time offloading mechanism tailored for dynamic and scalable edge environments. Our approach reformulates the offloading problem within a Semi-Markov Decision Process (SMDP) framework and introduces an adaptive optimization mechanism utilizing attention-based graph operations for heterogeneous Resource environments. This system, like how we prioritize tasks and divide resources, figures out how much attention to pay to each task and which server should handle it, to make things work smoothly. To make this work even better in the real world, we use a special method to adjust the rewards, which helps the system learn and improve its performance over time

1. Introduction

The rapid expansion of mobile networks and the proliferation of connected devices have transformed modern computing environments. From autonomous vehicles to immersive augmented reality applications, the demand for high-speed, low-latency services has surged. Traditional cloud computing architectures, despite their powerful centralized Resources, often fall short in meeting these latency-sensitive requirements due to long transmission distances and centralized processing bottlenecks [1]-[5]. This gap has driven the evolution of edge computing, which brings computation and storage closer to end-users by deploying edge servers (ESs) within the network's proximity. Within this paradigm, tasks may be executed locally or offloaded to nearby ESs. While ESs are equipped with more robust computational capabilities compared to UDs, the process of uploading tasks to ESs introduces additional energy consumption and latency. Moreover, the computational capacity of ESs remains constrained compared to centralized

cloud servers, making them unsuitable for handling large volumes of concurrent tasks. Resource contention among multiple tasks can degrade system performance and quality of service (QoS) [6], [7]. Consequently, devising an efficient scheduling mechanism for task offloading has become critical. Such mechanisms aim to optimize the selection of offloading targets and Resource allocation strategies [8], often framed as mixed-integer nonlinear programming (MINLP) problems, which are known to be NP-hard [9].
Initially, mathematical approaches [10] were developed to solve these optimization problems. However, these model-based methods struggle with generalization across diverse edge systems characterized by heterogeneous transmission technologies, application requirements, and computational Resources. To address this limitation, model-free metaheuristic algorithms [11, 12] were introduced for task offloading. Despite their flexibility, these algorithms face significant challenges, including large search spaces and poor adaptability to dynamic edge environments. In recent years, Deep Reinforcement Learning (DRL) has demonstrated exceptional capabilities across various domains, such as robotics control, autonomous driving, and natural language processing. Leveraging deep neural networks, DRL combines high-dimensional data analysis with model-free learning, making it a compelling choice for dynamic edge systems. Its online learning capabilities enable adaptive policy updates through continuous interaction with the environment, offering real-time adaptability to evolving edge conditions. As a result, DRL-based methods have shown promising results in optimizing task offloading and Resource allocation in edge computing [13]-[16]. Despite its advantages, DRL-based approaches face inherent limitations, as illustrated in Fig. 1. Firstly, these methods typically rely on discrete-time Markov Decision Processes (MDPs), where decisions are made at fixed intervals. This framework necessitates batch processing of tasks, causing delays as tasks wait for the next decision interval to be scheduled [17]. Such wait-for-scheduling latency increases Resource contention and lowers task completion rates, particularly in systems with stringent delay requirements. Secondly, traditional DRL methods lack scalability [18, 19]. The fixed computational graph of deep neural networks requires consistent input and output dimensions, making it challenging to adapt to varying system scales [20]. For instance, in mobile edge environments, the dynamic nature of vehicular edge systems—with frequent arrivals and departures of service or user vehicles—renders non-scalable DRL approaches infeasible. Retaining scalability under these conditions is crucial but often necessitates retraining models, a process that is both time-intensive and computationally expensive.

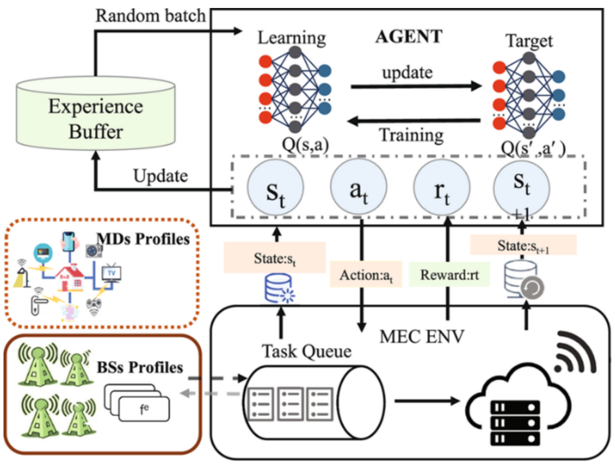


Figure 1. Challenges in DRL-Based Offloading Approaches.

Transitioning from a batched offloading framework to a real-time approach, where tasks are immediately scheduled upon arrival, intuitively minimizes waiting time and avoids dimensional mismatches caused by fluctuating task volumes. However, the discrete-time MDP framework utilized by classical DRL algorithms is inherently unsuitable for such scenarios [21]-[23]. Additionally, scalability challenges, such as mismatches in the dimensions of inputs and outputs caused by dynamic variations in the number of edge servers (ESs) and user devices (UDs), remain unresolved. To address these challenges, we propose a Real-time and Scalable Task Offloading framework (ReSTO), leveraging a DRL-based methodology.
In ReSTO, the task offloading problem is modeled as a Semi-Markov Decision Process (Semi-MDP) to enable decision-making at arbitrary task arrival times. The framework introduces the Scalable Continuous Proximal Policy Optimization (SCPPO) algorithm, specifically designed to align with the

Semi-MDP framework. To ensure scalability, SCPPO employs a heterogeneous graph attention mechanism for feature extraction, translating task-specific characteristics into adaptive attention scores for decision-making. Moreover, we develop a hybrid reward mechanism that integrates model-based and real-time feedback, referred to as the homotopy reward. This reward scheme bridges the gap between theoretical models and real-world dynamics while enhancing exploration efficiency during learning.

This paper aims to address the limitations of existing DRL-based task offloading approaches in edge computing environments. Specifically, we focus on:
1- Overcoming the limitations of discrete-time MDPs: We propose a novel continuous-time DRL framework that enables real-time, event-triggered task scheduling, eliminating the need for batch processing and reducing wait-for-scheduling latency.
2- Improving scalability in dynamic environments: We introduce a scalable DRL architecture that can
The key contributions of this work are as follows:

- **Introduction of ReSTO Framework:**

  We propose ReSTO, a novel real-time and scalable task offloading framework. ReSTO models the offloading problem using a Semi-MDP and introduces the SCPPO algorithm for real-time decision-making, eliminating the latency associated with traditional batched scheduling.
- **Scalability via Graph Attention Mechanism:**

  SCPPO employs heterogeneous graph attention operations to extract task and Resource features dynamically, enabling adaptive attention score generation. This approach prevents dimensional mismatches as the number of ESs or UDs changes, ensuring scalability.
- **Development of Homotopy Reward:**

  We formulate a hybrid reward system combining theoretical model rewards with real-time feedback. This homotopy reward reduces the disparity between theoretical assumptions and real-world conditions, improving both performance and exploration efficiency.

The remainder of this paper is organized as follows: Section II reviews related works, particularly focusing on real-time and scalable RL/DRL-based approaches. Section III presents the system model for real-time offloading and the corresponding optimization problem. In Section IV, we detail the ReSTO framework, including the Semi-MDP formulation and the SCPPO algorithm design. Section V evaluates ReSTO's performance

adapt to varying numbers of tasks and edge servers without requiring extensive model retraining.
By achieving these objectives, we aim to:

- Enhance task completion rates and reduce latency in edge computing systems with stringent performance requirements.
- Improve resource utilization by enabling more efficient task scheduling and allocation.
- Increase the adaptability and robustness of DRL-based offloading solutions in dynamic and unpredictable edge environments.

against state-of-the-art algorithms, highlighting its scalability and efficiency. Finally, Section VI concludes the paper with insights and potential future directions.

2. Related Works

In this section, we provide a comprehensive review of DRL-based task offloading methods. Following this, we delve into existing RL/DRL approaches for real-time or scalable task offloading, analyzing their achievements and limitations in comparison to our proposed framework.

A. DRL-Based Task Offloading in Edge Computing

Over the past decade, task offloading in edge computing systems has increasingly relied on Deep Reinforcement Learning (DRL) algorithms due to their capacity for dynamic decision-making and adaptability to complex environments. These algorithms leverage the ability of neural networks to process high-dimensional inputs and learn optimal policies directly through interaction with the environment. Numerous studies have tailored DRL methods to address the unique challenges of edge systems, such as Resource constraints, latency requirements, and dynamic user demands. One notable example is the work of Wang et al. [12], who utilize Deep Q-Learning (DQN) to optimize both task offloading and Resource configuration in a blockchain-enabled edge computing framework. Their approach introduces trust mechanisms and leverages blockchain for secure and efficient offloading. Similarly, Huang et al. [13] employ a Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm for partial offloading systems, where tasks can be split between local and edge processing. This method improves decision-making by accounting for the variability in task size and Resource availability, demonstrating the potential of DRL in adaptive task allocation.

Building on these foundational approaches, subsequent research has focused on enhancing the performance and robustness of DRL-based task offloading. For instance, Xu et al. [14] and Ma et al. [15] introduce temporal feature extraction to capture the dynamic nature of edge environments, utilizing historical state information to better model system behavior and predict the effects of various actions. This temporal awareness allows the system to adapt to changing workloads and network conditions, leading to more effective offloading strategies.

Moreover, Xu et al. [16] propose an exploration-exploitation strategy tailored to the training process. By prioritizing exploration during the early stages of training and gradually shifting towards the exploitation of learned policies, their approach strikes a balance between discovering new solutions and refining existing ones. This adaptive strategy improves policy performance and ensures more reliable decision-making over time.

To address the computational complexity and convergence challenges associated with large action spaces, researchers have also explored hybrid approaches that integrate DRL with traditional optimization techniques. For example, Chen et al. [17] enhance DQN-based task offloading with sequential quadratic programming for Resource allocation. This combination reduces the dimensionality of the problem and accelerates convergence, enabling more efficient use of edge Resources.

Li et al. [18] take a multi-agent approach, employing a Parameterized Multi-Agent Soft Actor-Critic (SAC) algorithm to address the interdependence of actions across agents. By categorizing actions into those that affect other agents and those that do not, they effectively manage Resource contention in collaborative edge environments. The use of a genetic algorithm further refines Resource allocation decisions, ensuring optimal system performance.

Despite these advancements, existing DRL-based methods face inherent limitations due to their reliance on the discrete-time Markov Decision Process (MDP) framework. This framework enforces decision-making at fixed intervals, leading to batch processing of tasks. Such a structure introduces scheduling delays, as tasks must wait until the next decision point before offloading can occur [24], [25]. This wait-for-scheduling latency becomes particularly problematic in latency-sensitive applications, where even slight delays can significantly degrade performance. Additionally, most DRL approaches encode system states into a one-dimensional input vector for processing by a multi-layer perceptron (MLP). While this design simplifies implementation, it limits scalability. Fixed input-output dimensions in MLPs cannot adapt to changes in the number of edge servers (ESs) or user devices (UDs), resulting in dimensional mismatches. This lack of flexibility hampers the applicability of DRL algorithms in dynamic edge environments, such as vehicular networks or large-scale IoT systems, where the network topology and Resource availability frequently change.

These challenges underscore the need for novel frameworks and algorithms that overcome the constraints of discrete-time MDPs and enable real-time, scalable task offloading in edge computing systems. Future solutions must address both the latency introduced by batch processing and the scalability issues arising from static neural network architectures, paving the way for more adaptive and efficient DRL applications in edge environments.

- Categorization by Objective:
  1. Latency Minimization: Focus on methods specifically designed to minimize task completion time or end-to-end delay.
  2. Energy Efficiency: Analyze methods that prioritize minimizing energy consumption at the device and network levels.
  3. Resource Allocation: Discuss approaches that optimize resource allocation among UDs and ESs, considering factors like CPU, memory, and bandwidth.
  4. Load Balancing: Examine methods that aim to distribute the computational load evenly across the available ESs.

B. Real-Time RL/DRL for Task Scheduling

Real-time decision-making is a critical component of task scheduling in edge computing and numerous other domains, where rapid responses to dynamic changes are essential for maintaining system performance and efficiency. However, the discrete-time Markov Decision Process (MDP) framework, which underpins most traditional RL/DRL methods, introduces inherent constraints when applied to real-time applications. By requiring fixed decision intervals, the discrete-time MDP framework creates bottlenecks, such as delays in task execution, that compromise the responsiveness and adaptability of RL-based solutions. Alternative frameworks, such as the multi-armed bandit [26]-[30], have been explored to address some of these challenges. While these models are computationally simpler and focus on optimizing immediate rewards, they often fail to account for the temporal dependencies and cumulative effects of actions. This omission can lead to suboptimal decision-making, particularly in complex and dynamic environments where long-term outcomes must be carefully balanced with short-term gains [31]-[33].

In contrast, the Semi-Markov Decision Process (Semi-MDP) framework is particularly well-suited for real-time scheduling tasks. Unlike the discrete-time MDP, Semi-MDP allows for variable intervals between decision points, making it more flexible and capable of handling tasks as they arrive. This flexibility enables the development of policies that optimize long-term performance while addressing the immediate requirements of real-time systems. For instance, Liang et al. [20] and Hao et al. [21] successfully use Semi-MDPs to model real-time scheduling problems, demonstrating the framework's potential to accommodate dynamic workloads and varying system conditions. Despite its advantages, adapting existing algorithms to the Semi-MDP framework poses unique challenges due to its structural differences from the traditional MDP approach. One common strategy involves normalization, which converts Semi-MDP problems into an MDP-compatible format, allowing established DRL algorithms to be applied. For example, Liang et al. [22] normalize Semi-MDP problems by estimating theoretical model-based Q-values for supervised pre-training [34]-[36]. This approach provides a starting point for the policy, which is then refined through interactions with the environment. Similarly, Wu et al. [23] utilize state transition probabilities during the normalization process to transform Semi-MDPs into a form solvable by value iteration techniques.

An alternative to normalization-based methods is the direct design of algorithms tailored to the Semi-MDP framework. These approaches avoid the approximations and assumptions inherent in normalization, enabling more accurate modeling of real-world scenarios. For example, Van Huynh et al. [24] propose a Dueling Double Deep Q-Network (DDQN) approach that maximizes cumulative single rewards without incorporating discount factors, focusing instead on immediate benefits within a Semi-MDP structure. Wei et al. [9] employ an exponential decay model to compute cumulative discounted returns, deriving a Bellman optimality equation to guide decision-making with DQN. Kim et al. [25] adapt the Soft Actor-Critic (SAC) algorithm for the Semi-MDP framework, introducing modifications that account for the variable time intervals and cumulative reward structures characteristic of Semi-MDPs. Despite these advancements, existing methods still exhibit notable limitations. Normalization-based approaches often rely heavily on theoretical assumptions, such as idealized transition models or fixed state representations, which reduce their generalizability to real-world, complex environments [37]-[40]. These assumptions can lead to performance degradation when applied to heterogeneous and highly dynamic edge systems, where practical constraints and unpredictable factors frequently deviate from theoretical models. On the other hand, model-free DRL approaches [41]-[45] that bypass theoretical dependencies also face challenges. These methods commonly employ simplistic neural network architectures, such as basic feedforward models, that lack the scalability needed to adapt to dynamic edge network conditions. In systems where the number of edge servers (ESs) and user devices (UDs) can fluctuate significantly, fixed input-output dimensions lead to dimensional mismatches, requiring costly retraining of the models to accommodate changes [46]-[48]. This inflexibility limits the practical deployment of model-free DRL solutions in scenarios characterized by high variability and evolving system requirements. Overall, while the Semi-MDP framework offers significant potential for enabling real-time decision-making in edge computing, achieving effective and scalable solutions necessitates innovative algorithmic designs that address both the limitations of normalization-based methods and the scalability constraints of traditional DRL models. Future work must focus on bridging these gaps to develop robust and adaptable frameworks capable of supporting real-time, scalable task scheduling in edge environments.

- Weaknesses of Current Semi-MDP Methods:

1. Normalization-Based Approaches:
2. Reliance on Theoretical Assumptions: Often rely on idealized models and assumptions, which can limit their applicability in real-world scenarios with high variability and uncertainty.
3. Potential for Accuracy Loss: The normalization process can introduce approximations that may lead to

suboptimal solutions or reduced accuracy.

4. Limited Exploration of Direct Semi-MDP Algorithms: While some direct approaches exist, the field is still relatively under-explored compared to normalization-based methods.
5. Scalability Challenges: As the complexity of the environment and the number of tasks increase, solving Semi-MDPs can become computationally expensive, especially for complex DRL algorithms.
6. Handling of Uncertainty: Many existing methods may not adequately address the inherent uncertainty and stochasticity present in real-world scheduling problems.

3. System Model and Problem Formulation

We consider a crowdsourcing-inspired MEC system, as illustrated in Fig. 1, comprising multiple applications and edge servers (ESs) with diverse configurations and characteristics. These applications may vary significantly in their requirements, encompassing delay-sensitive services such as networked gaming, autonomous driving, and AR/VR, as well as resource-intensive tasks like big data analytics, scientific computing, and video surveillance [49]. Similarly, ESs can range from micro data centers and edge clouds to high-capacity computing servers or even gateways deployed in residential or office settings. For generality, we assume these ESs are managed and operated by distinct edge service providers. To maximize resource utilization and enhance system performance in terms of scalability, reliability, and other metrics, a third-party platform is introduced to coordinate ES operations and handle workload dispatch from end users. Acting as an intermediary, this platform serves as a front-end interface for edge computing services, bridging the gap between clients submitting tasks and ESs providing computational resources. Upon receiving a task, the platform assigns it to the most suitable ES hosting the requested service and ensures the computation result is returned to the client seamlessly. This interaction is transparent to users, provided the system meets their application performance expectations, such as low latency and high computation quality.

Both application providers and ESs must undergo an onboarding process with the platform before accessing or delivering edge services. This formalized process involves signing agreements with the platform to define roles and responsibilities. For application providers, this includes specifying service requirements such as task rates, task valuation, budget constraints, computational demands, QoS parameters (e.g., maximum tolerable delay), and security or compliance needs. Similarly, ESs seeking to participate in the system are subject to a comprehensive evaluation by the platform. This involves reviewing their security protocols, compliance certifications, and data management practices to ensure adherence to industry standards and regulatory requirements [50]. Additionally, a risk assessment is often conducted to identify potential vulnerabilities. ESs must provide detailed information regarding their resource capacities, operational costs, and revenue expectations.

Using this information, the platform optimizes task offloading strategies and resource allocation for ESs, subsequently formalizing agreements with both parties. Once agreements are in place, ESs configure the necessary accounts and infrastructure, enabling application providers to deploy their services. Importantly, ongoing monitoring and auditing mechanisms are established to ensure all parties adhere to the agreed-upon terms, with regular performance and compliance evaluations conducted throughout the service lifecycle.

This study considers a scenario where application providers make advance payments to the platform, which, in turn, allocates a portion of these payments to incentivize contributions from edge servers (ESs). The platform's key decisions include: (1) whether to accept both the application providers and ESs into the system, (2) determining the amount of resources each ES should allocate to applications, and (3) devising an efficient task dispatching strategy to distribute tasks among the backend ESs hosting the services. To simplify notation, we define the set of ESs and applications/services in the system as M and N, respectively, with the corresponding cardinalities denoted by |M| and |N|. For clarity, the terms "applications" and "application providers" are used interchangeably in this paper unless otherwise specified. The primary notations employed throughout this work are summarized in Table 1.

Each application $i \in N_i$ is characterized by a tuple $(p_i, v_i, \alpha_i, D_i, s_i)$,where:

1. p_i: The payment made by application provider iii to the platform for task offloading.

2. v_i: The utility gained by i from offloading a task, such as reduced energy consumption at user devices, enhanced computational quality, or shorter response times. Generally, $p_i \leq v_i$ Offloading offers net benefits to the application.
3. α_i: The arrival rate of tasks for application i.
4. s_i: The workload (measured in CPU cycles) required to process a task.
5. D_i: The maximum latency tolerable by application i.

Given the stochastic nature of the system and the uncertainty in resource allocation at ESs, the actual value derived by an application from task offloading depends on the quality of the edge computing service. We represent this with a utility function $u_{ij} \in [0,1]$, which quantifies the satisfaction level of application i when offloading tasks to ES_j. This utility function is an abstract representation and can vary depending on the application's requirements.
For instance, for delay-sensitive applications, u_{ij}may be defined based on reductions in task latency. For resource-intensive applications, u_{ij}could reflect the computational quality, such as compression ratios or prediction accuracy. Moreover, the form of u_{ij} can differ even within the same application category. For example, in delay-sensitive applications, u_{ij} could be a step function to model satisfaction levels in the presence of hard deadlines.

$$u_{ij} = \begin{cases} 1, & \text{if } \ t_{ij} \leq D_i \\ 0, & \text{otherwise} \end{cases} \qquad (1)$$

A. Platform Model

The platform operates under the following assumptions:

1. The platform employs a probabilistic task dispatching mechanism, where each application task is routed to a specific ES based on predefined probabilities.
2. The payment p_i made by application iii is distributed between the platform and the ES executing the task. Specifically, the ES receives a reward of $(1-\lambda_i)p_i$, where $\lambda_i \leq 1$, while the platform retains $\lambda_i p_i$ as its service charge or maintenance fee. The parameter λ_i , a critical system variable, is determined by the platform and forms part of the contractual agreement with the ES.

4. Real-time and Scalable Task Offloading Framework

Before detailing the algorithm, we first describe the calculation of u_{ij} and t_{ij} under a fixed resource allocation $F_{ij} = F_0$. The following assumptions, drawn from prior studies, are applied:

1. Tasks from each application arrive according to a Poisson process [46]. Consequently, the arrival of tasks from application i at ES_j also follows a Poisson process with a rate of $r_{ij} = \alpha_i x_{ij}$, where α_i represents the task arrival rate, and x_{ij} denotes the probability of task dispatch to ES_j .

2. The workload of tasks from each application is assumed to follow an exponential distribution (in CPU cycles) [27][36]. This implies that the processing time for a task from application i at ES_j also follows an exponential distribution with a mean of 1/wij1/w where $w_{ij} = F_{ij}(0)$ and s_i represents the workload of the task.

Based on these assumptions, the task processing system for an application i at ES_j can be modeled as an M/M/1queue. The probability density function (pdf) for the task delay t_{ij} this system is then expressed as:

$$f_T(t_{ij} \leq t) = (w_{ij} - r_{ij}) \cdot e^{-(w_{ij}-r_{ij})t} \qquad (2)$$

Assuming u_{ij} is defined as in Eq. (2) and $x_{ij} > 0$ (indicating that tasks from application i are offloaded to ES_j), the relationship derived from constraint (3b) is as follows:

$$\Pr\left(t_{ij} < \left(1 - \tfrac{p_i}{v_i}\right) D_i\right) \geq \text{prob}_i$$
(3)

Combining (7) and (8), we get:

$$x_{ij} \leq \frac{1}{\alpha_i}\left[\frac{\ln(1-prob_i)}{\left(1-\frac{p_i}{v_i}\right)D_i} + \frac{F_{ij}^0}{s_i}\right] \qquad (4)$$

Let $x_{ij}H_x$ denote the right-hand side (RHS) of the inequality mentioned above, defined as:

$$x_{ij}^{H} \triangleq \frac{1}{\alpha_i}\left[\frac{\ln(1-prob_i)}{\left(1-\frac{p_i}{v_i}\right)D_i}+\frac{F_{ij}^{0}}{s_i}\right] \qquad (5)$$

Clearly, $x_{ij}H_x$ represents the upper bound of the offloading probability for which application provider i is satisfied with offloading its tasks to ES_j , meeting the QoS requirements. Notably, this upper bound is independent of λ_i and is solely determined by F_{ij} (0) and the workload profiles.

Similarly, from constraint (3c) and assuming $x_{ij} > 0$, we derive:

$$x_{ij} \geq \frac{(1+\beta_{ji})c_j\left(F_{ij}^{0}\right)}{\alpha_i(1-\lambda_i)p_i}, \qquad (6)$$

Let the right-hand side (RHS) of the above inequality be denoted as x_{ij}^{L} , defined as:

$$x_{ij}^{L} \triangleq \frac{(1+\beta_{ji})c_j\left(F_{ij}^{0}\right)}{\alpha_i(1-\lambda_i)p_i}. \qquad (7)$$

Algorithm 1 Deriving the optimal resource allocation, task offloading probabilities, and ratios under a given resource allocation $F_{ij}^{0\prime}$s.

```
   Input: Task profiles ($\alpha_i's, p_i's, v_i's, D_i's, s_i's$); ES profiles ($c_j(F_{ij})'s, F_j's, \beta_{ji}'s$); Initial resource allocations $F_{ij}^{0\prime}s$;
   Output: Resource allocations $F_{ij}^{0,\sim\prime}s$; Ratios $\lambda_i^{0,\sim\prime}s$; Task offloading probabilities $x_{ij}^{0,\sim\prime}s$;
1  for $i \in \mathcal{N}$ do
2      for $j \in \mathcal{M}$ do
3          Derive $x_{ij}^{H}$ and $\lambda_{ij}^{0}$ according to Eq 5.
4  for $i \in \mathcal{N}$ do
5      Get $\lambda_i^{0}$ and $x_{ij}^{0}$
6  for $j \in \mathcal{M}$ do
7      Get $\mathcal{Y}_{ij}^{0}$ ́s
8  Obtain $\lambda_i^{0,\sim}$ ́s, $F_{ij}^{0,\sim}$ ́s and $x_{ij}^{0,\sim}$ s ́
```

5. Simulation Experiments

A. Experimental Setup

The simulation framework was developed using Python 3.9 and PyTorch 2.3.0, running on a high-performance desktop system powered by an Intel Core i9-13900K processor and an Nvidia GeForce RTX 3090 GPU. This computational setup was chosen to ensure efficient processing of the complex algorithms and large-scale datasets involved. The simulation leverages vehicle trajectory data from the Peachtree Street section of the Next Generation Simulation (NGSIM) dataset [36]. This dataset provides detailed and realistic representations of urban traffic flow, making it suitable for modeling dynamic user-device behaviors in edge computing scenarios.

In our simulation environment, user devices (UDs) are designed to move along stochastic trajectories generated from the NGSIM dataset. These trajectories simulate real-world mobility patterns, such as vehicles traveling through a busy metropolitan area. UDs are assumed to exit the system once their respective trajectories conclude, reflecting the dynamic entry and exit behavior typical in edge networks. Edge nodes (ENs) are deployed strategically at random locations along these trajectories, ensuring adequate coverage of user mobility patterns while capturing the inherent randomness of real-world deployments. The system parameters used in the simulation are comprehensively detailed in Table II. These include network characteristics, Resource configurations, and mobility patterns, ensuring that the simulation accurately reflects the operational constraints and requirements of modern edge computing environments.

Training Process and Network Design

The training process was meticulously designed to optimize the learning performance of the proposed algorithm. The neural network architecture incorporates several specialized components to handle the complexity of real-time task offloading and Resource allocation. The hidden feature dimension d was set to 256, balancing computational efficiency with model expressiveness. The attention mechanism employed K=4 attention heads, enabling the model to capture intricate relationships between tasks and edge nodes across multiple dimensions.

Three encoder components—H_{EN}, H_{Cell}, and H_{Task} —were implemented as two-layer multilayer perceptrons (MLPs), each employing Tanh activation functions. These encoders transform raw input data into high-dimensional representations suitable for downstream processing. The MLP Dc, responsible for computing Resource allocation, was configured with two layers, ensuring lightweight and efficient computation. In contrast, the MLP Dv within the critic network was designed with four layers to enhance its capacity for estimating value functions, which are critical for effective policy evaluation and improvement.

Key Parameters for the Continuous-Time PPO Algorithm

To align the training process with the Semi-Markov Decision Process (Semi-MDP) framework, we tailored the continuous-time Proximal Policy Optimization (PPO) algorithm with carefully selected hyperparameters. The discount factor α was set to 0.1, ensuring a balanced emphasis on immediate rewards and long-term gains. The importance sampling ratio ϵ, set to 0.2, controlled the degree of policy updates to maintain stability during training. The Generalized Advantage Estimation (GAE) hyperparameter λ was configured as 0.98 to improve the estimation of advantages, enhancing the convergence rate and overall learning efficiency.

B. Training Configuration and Iterations

The training process spanned 400 episodes, providing sufficient iterations for the algorithm to converge to an optimal policy. Each episode was further divided into a maximum of 200 iterations, allowing the model to explore diverse states and actions comprehensively. During training, the model continually interacted with the simulated environment, refining its policy through trial and error while leveraging feedback from the homotopy reward mechanism. This hybrid reward system combined theoretical insights with real-time observations, bridging the gap between simulated models and practical deployments.
The overall design of the simulation environment, coupled with the robust training setup, ensures that the proposed algorithm is well-equipped to handle dynamic and scalable edge computing scenarios. By incorporating realistic mobility patterns, stochastic task generation, and advanced neural network architectures, the simulation framework provides a reliable foundation for evaluating the effectiveness of real-time task offloading and Resource allocation strategies in next-generation edge systems.

TABLE I.
PARAMETER SETTINGS OF SIMULATION

Notations	**Simulation Value**	**Notations**	**Simulation Value**
M	8	α	U(1.0, 1.2) MB U (0.8, 1.0) GCycle U (1, 2) Second
$\boldsymbol{f_m}$	U (2, 4) GHz	β	
$\boldsymbol{d^m}$	50 Meter	ϑ	
N	30	p	
			1 Watt
$\boldsymbol{q^{max}}$	3	ς	-3
$\boldsymbol{\iota}$	1	σ^2	-114 dBm/MHz
X	0.1	B	1 MHz
$\boldsymbol{f_n}$	U (1, 2) GHz	**Ω**	1
ϒ	4	K	10^{-27}

The data reuse frequency was configured to 10 iterations. For the actor-network, the learning rate was set to 1×10^{-4}, while the critic network utilized a higher learning rate of 1×10^{-3}.The Adam optimizer, with $\varepsilon = 1 \times 10^{-5}$,was employed for parameter updates.
To evaluate the performance of the proposed method, we conducted a comparative analysis with four state-of-the-art DRL-based methods designed to address scalability, as well as a single-step greedy method. A brief overview of these approaches is as follows:

- **Single-Step Greedy (SSG):** This method selects actions greedily based on immediate task benefits. While intuitive, it focuses exclusively on short-term gains, neglecting long-term system optimization.
- **Sequence to Sequence (S2S) [11]:**
  This approach leverages recurrent neural networks (RNNs) for sequential system feature extraction and multi-action generation. However, it operates under a batched offloading framework and struggles to adapt action dimensions to dynamic variations in the number of edge nodes (ENs).
- **Self-Attention (SA) [10]:**
  Using a self-attention mechanism, this method integrates task features and generates actions in parallel. Despite this, it inherits the limitations of S2S, including reliance on batched offloading and the inability to adapt to changes in EN counts due to its concatenation of EN states as input.
- **Event-Driven DQN (EDQ) [9]:**
  This real-time approach employs an event-driven Deep Q-learning framework based on task and EN states. However, its reliance on a multilayer perceptron (MLP) architecture for the Q-network constrains scalability, particularly in large-scale systems.
- **GNN-based Multi-agent DRL (GMD) [30]:**
  This method utilizes a distributed multi-agent DRL framework with graph neural

networks (GNNs), allowing user devices (UDs) to independently select actions. By representing offloading targets as positive integers instead of one-hot vectors, it offers significant scalability. However, multi-agent DRL frameworks are challenging to train in large-scale environments, often leading to diminished performance.

For a fair comparison, we set the batch interval to 0.2 in subsequent experiments for the S2S, SA, and GMD methods, which follow a batched offloading framework.

Notably, the ReSTO framework outperformed all baselines in terms of system cost, even under zero-shot transfer scenarios, surpassing re-trained methods as well. This underscores the exceptional scalability and efficiency of ReSTO. Interestingly, we observed that the system costs of SA and EDQ remained stable or even increased as additional ENs became available. This phenomenon is attributable to their reliance on concatenated EN states as input, which inflates the input dimensions, causing the critic network to struggle with accurate evaluations. For EDQ, the increase in selectable actions further complicates Q-network convergence, exacerbating its limitations in larger systems.

C. Batched Offloading V.S. Real-Time Offloading

To highlight the performance benefits of transitioning from batched offloading to real-time offloading, we compare the proposed ReSTO method with existing approaches under two load scenarios. The results, as illustrated in Fig. 2, consider a normal scenario with baseline system settings and a harsh scenario where the load factor $\beta \in u(1.2,1.4)$. For consistency, we introduce artificial delays in task execution to emulate batched offloading for ReSTO, SSG, and EDQ, which inherently support real-time offloading. Other methods, lacking real-time capabilities, are excluded from this analysis. Batched offloading is tested with four discrete timeslot intervals: 0.8, 0.6, 0.4, and 0.2.

The experimental findings indicate that reducing the interval duration in batched offloading substantially lowers system costs under both load scenarios, with the real-time offloading approach consistently achieving the best performance. This improvement is especially pronounced under higher load conditions, as shorter decision intervals minimize the delay between task arrival and scheduling, allowing for more effective Resource management. Conversely, under increased system loads, extended waiting periods in batched offloading sharply reduce the scope for scheduling adjustments, leading to greater performance degradation. Notably, at elevated load levels with larger timeslot intervals, DRL-based methods display inferior performance compared to the SSG approach. This can be attributed to challenges in learning from delayed and sparse rewards during training, particularly when task failures dominate the early learning phase. As a result, many DRL-based methods converge to suboptimal solutions, unable to recover effectively. In contrast, the ReSTO framework, supported by the homotopy reward mechanism, provides more immediate and structured reward feedback during early training stages. This design facilitates more efficient exploration and allows ReSTO to avoid local optima, delivering significantly better performance even under harsh conditions.

D. Ablation Study

An ablation study was conducted to investigate the impact of the homotopy reward design and graph-based cell state aggregation on the performance of the proposed framework. The experiments were carried out under both normal and harsh scenarios to provide a comprehensive evaluation across varying load levels. Two key components were evaluated: (1) the reward mechanism, with three configurations considered—model-based reward, reality reward, and the proposed homotopy reward—and (2) the user device (UD) state fusion method, comparing direct aggregation of UD states independently versus graph-based aggregation of cell states. These configurations were systematically combined into multiple algorithm variants, and their performance was assessed.

The study revealed significant differences in performance across the reward settings. Among the configurations, the reality reward (blue line) exhibited the largest fluctuations during training. These fluctuations can be attributed to the reward mechanism's reliance on real-time feedback, which is inherently noisy and less predictable. The lack of robust guidance in the early training stages often led to instability in task success rates, particularly under harsh scenarios where Resource constraints are more pronounced. Additionally, this configuration struggled to balance immediate performance with long-term optimization, highlighting its limitations in dynamic and unpredictable environments.

Conversely, the model-based reward demonstrated greater stability but was less effective in capturing the complexities of real-world conditions. This resulted in suboptimal exploration, limiting its ability to adapt to diverse scenarios. The proposed homotopy reward bridged the gap between the model-based and reality rewards, effectively integrating theoretical guidance with real-time feedback. This hybrid approach significantly improved exploration efficiency, enabling the algorithm to converge faster and achieve better performance across both normal and harsh scenarios. The homotopy reward design also mitigated the challenges of sparse rewards, ensuring consistent progress during training.

The study further examined the effects of state aggregation methods. Directly aggregating UD states independently often resulted in subpar system performance due to the lack of contextual understanding of Resource and task interactions within the network. In contrast, the graph-based cell state aggregation effectively captured spatial and temporal dependencies, enhancing the framework's ability to adapt to changes in system dynamics. By leveraging graph structures to model interactions between tasks and edge servers (ESs), this method provided a holistic view of the network, leading to more informed and efficient decision-making.

The analysis also sheds light on the limitations of the GMD algorithm, which demonstrated a tendency to prioritize tasks with higher energy consumption. This behavior can be traced to its distributed multi-agent DRL framework, where each agent operates with limited visibility into the overall system state. Without a comprehensive view of the network, agents often opted to process tasks at a higher frequency to minimize CPU occupancy and avoid Resource contention. While this strategy may reduce immediate delays, it inadvertently increases energy consumption and diminishes the overall system efficiency.

In summary, the results highlight the advantages of the proposed homotopy reward design and graph-based cell state aggregation in improving system performance and scalability. By addressing the shortcomings of traditional reward mechanisms and state aggregation methods, the proposed approach achieves superior stability, faster convergence, and enhanced adaptability, particularly under challenging operational conditions.

E. Comparisons under Different Environmental Settings

This section evaluates the performance of our proposed algorithm against other methods under varying simulation parameters, specifically focusing on the task generation interval parameter (Ω) of the exponential distribution and the user preference for required CPU cycles $\overline{\beta}$). These parameters influence the system load by altering the task arrival rate and the computational demand of each task. Our analytics illustrate the system costs across different values of Ω. A reduction in Ω corresponds to an increased number of tasks and a heavier overall system load. The results reveal that DRL-based methods consistently outperform the SSG approach in all scenarios. This is due to the long-term optimization capabilities inherent in DRL, which enable proactive and foresight-driven decision-making. In contrast, the SSG method prioritizes immediate task optimization without accounting for future system demands, leading to significant queue delays and higher overall costs.

Among the DRL-based methods, the S2S approach exhibits comparatively higher system costs. This can be attributed to its vulnerability to the memory-forgetting issue associated with processing long task sequences. As Ω decreases, the number of tasks requiring scheduling within each discrete timeslot increases, further amplifying this limitation. In contrast, the proposed ReSTO framework achieves the lowest system cost across all scenarios, with the performance gap widenthe ing as Ω decreases. This superior performance stems from the fundamental differences between real-time and batched offloading. As the system load intensifies with a higher task arrival rate, the limitations of batched offloading become more pronounced, leading to greater performance degradation for methods relying on discrete scheduling intervals. These findings reaffirm the advantages of the real-time offloading strategy employed in ReSTO, particularly under high-load conditions.

Our analytics compares the performance of the algorithms across different values of $\overline{\beta}$, which represents the computational load associated with tasks. Higher $\overline{\beta}$ values indicate that tasks demand more CPU cycles for processing, thereby increasing the system load. The results reveal that under low-load scenarios, DRL-based methods demonstrate a clear advantage over the Single-Step Greedy (SSG) approach, achieving significantly lower system costs. This improvement is attributed to the long-term optimization capabilities of DRL, which enable more efficient Resource allocation and task scheduling by anticipating future system states. In contrast, SSG focuses solely on immediate task optimization, often resulting in suboptimal Resource utilization and increased queuing delays. As the system load intensifies with higher $\overline{\beta}$ values, the performance gap between

DRL-based methods and SSG narrows. This reduction in effectiveness stems from the challenges introduced by the more demanding environment. Heavier system loads generate delayed and sparse rewards, complicating the training process for DRL algorithms and limiting their ability to converge to optimal policies. Under these conditions, traditional DRL-based approaches are more likely to become trapped in local optima, as the sparse feedback makes it difficult to identify and reinforce effective scheduling strategies.

The proposed ReSTO framework, however, addresses these limitations through its innovative homotopy reward mechanism. By combining model-based and reality-based rewards, the homotopy reward provides consistent and structured feedback throughout the training process. This design enables ReSTO to navigate complex and dynamic system states more effectively, avoiding local optima and guiding the algorithm toward globally optimized solutions. The ability to adapt to varying load conditions is further enhanced by the real-time offloading strategy employed in ReSTO, which eliminates the delays associated with batched scheduling. This combination of timely decision-making and robust reward feedback allows ReSTO to maintain superior performance across all load conditions. Moreover, the advantages of ReSTO become increasingly pronounced as the system load rises. In high-load scenarios, where tasks require significant computational Resources and delays are more detrimental, the benefits of real-time offloading are particularly evident. By reducing the waiting time between task arrival and execution, ReSTO not only minimizes queuing delays but also maximizes Resource utilization efficiency. These factors collectively contribute to ReSTO's consistent outperformance of competing methods, demonstrating its scalability, adaptability, and resilience under diverse operational conditions.

In summary, the integration of the homotopy reward mechanism and real-time offloading in ReSTO provides a significant edge over existing DRL-based approaches and heuristic methods like SSG. The framework's ability to maintain low system costs under both low and high system loads highlights its robustness and makes it a promising solution for real-time and scalable task offloading in dynamic edge computing environments.

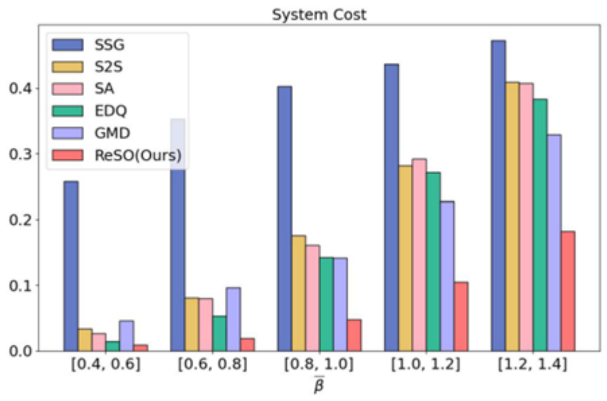


Fig 2. System Costs Across Algorithms for Varying Task CPU Cycle Requirements.

6. Conclusions

While DRL-based algorithms have demonstrated exceptional capabilities in optimizing task offloading for edge computing, several persistent challenges limit their potential for broader practical deployment. Key among these is the waiting time associated with batched decision-making and the dimensional mismatches arising from dynamic system scales. These limitations not only impede performance improvements but also hinder the scalability and adaptability of such methods in real-world applications. To address these critical issues, we introduce ReSTO, a DRL-driven real-time and scalable offloading framework designed to overcome the inherent challenges of existing methods. ReSTO redefines the task-offloading paradigm by shifting from a batched scheduling approach to a real-time offloading framework. Tasks are scheduled immediately upon arrival, eliminating waiting times and enabling more efficient Resource utilization. This is achieved by modeling the offloading problem as a Semi-Markov Decision Process (Semi-MDP), allowing decision-making at arbitrary task arrival times rather than fixed intervals. To effectively solve the problem, ReSTO employs a novel continuous-time Proximal Policy Optimization (PPO) algorithm, enhanced with specially designed scalable actor and critic networks that adapt seamlessly to varying numbers of edge nodes (ENs) and user devices (UDs). This architecture ensures robust performance across dynamic system conditions.

In addition to its innovative decision-making framework, ReSTO introduces two key mechanisms to further enhance its performance. First, the homotopy reward mechanism integrates model-based and reality-based rewards to bridge the gap between theoretical assumptions and real-world dynamics. This approach improves learning efficiency, enabling the algorithm to avoid local optima and converge toward globally optimal policies. Second, ReSTO clusters UDs into cells, aggregating state information to reduce dimensional complexity and improve decision accuracy. This clustering approach ensures scalability and effective Resource allocation even in large-scale systems with high task loads. Extensive experimental evaluations highlight the significant advantages of ReSTO over state-of-the-art algorithms. The results demonstrate that ReSTO consistently achieves lower system costs while exhibiting better scalability as the number of ENs and UDs fluctuates. These findings underscore the robustness and adaptability of the proposed framework, making it well-suited for the dynamic and heterogeneous environments characteristic of modern edge computing systems. However, transitioning from batch to real-time offloading also brings new challenges, particularly in terms of the computational overhead associated with state acquisition and decision-making processes. The need for rapid, real-time decisions places greater importance on minimizing time complexity to ensure the practical viability of ReSTO in large-scale deployments. Future work will focus on exploring and developing algorithms with reduced time complexity, capable of operating under partially updated or approximate state information. By addressing these challenges, we aim to further enhance the efficiency and scalability of real-time offloading solutions, paving the way for their widespread adoption in edge computing.

- Experimental Results and Validation:

Extensive simulations demonstrate the superior performance of ReSTO compared to state-of-the-art methods. Specifically, ReSTO consistently achieves lower system costs (e.g., energy consumption, latency) while exhibiting better scalability as the number of ENs and UDs fluctuates. These results validate the effectiveness of ReSTO in optimizing resource allocation and adapting to dynamic system conditions.

Conceptual Explanations:

- Addressing Batching Limitations: By moving to a real-time framework, ReSTO eliminates the inherent delay associated with batched decision-making, leading to more responsive and efficient resource allocation.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] Mirzaei, A. and Najafi Souha, A., 2021. Towards optimal configuration in MEC Neural networks: deep learning-based optimal resource allocation. Wireless Personal Communications, 121(1), pp.221-243.

[3] Zhou, Guoliang, and Amin Mohajer. "Blind reconfigurable intelligent surfaces for dynamic offloading in fixed-NOMA mobile edge networks." International Journal of Sensor Networks 46, no. 3 (2024): 142-160.

[4] H. Guo, J. Li, J. Liu, N. Tian, and N. Kato, “A survey on space-airground- sea integrated network security in 6g,” IEEE Communications Surveys & Tutorials, vol. 24, no. 1, pp. 53–87, 2022.

[5] Duan, H., & Mirzaei, A. (2023). Adaptive Rate Maximization and Hierarchical Resource Management for Underlay Spectrum Sharing NOMA HetNets with Hybrid Power Supplies. Mobile Networks and Applications, 1-17.

[6] Zhou, Nan, Ya Nan Li, and Amin Mohajer. "Distributed capacity optimisation and resource allocation in heterogeneous mobile networks using advanced serverless connectivity strategies." International Journal of Sensor Networks 45, no. 3 (2024): 127-147.

[7] X. Huang, Y. Chen, J. Liu, M. Wang, P. Li, and Q. Zhao, “Joint interdependent task scheduling and energy balancing for multi-uav enabled aerial edge computing: A multi-objective optimization approach,” IEEE Internet of Things Journal, vol. 10, no. 4, pp. 3147–3160, 2023.

[8] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, “Energy efficient Resource allocation in uav enabled mobile edge computing networks,”IEEE Transactions on Wireless Communications, vol. 18, no. 9, pp. 4576–4589, 2019.

[9] Mohajer, Amin, Mohammad Yousefvand, Ehsan Noori Ghalenoo, Parviz Mirzaei, and Ali Zamani. "Novel approach to sub-graph selection over coded wireless networks with QoS constraints." IETE Journal of Research 60, no. 3 (2014): 203-210.

[10] X. Zhang, J. Zhang, J. Xiong, L. Zhou, J. Wei, and H. Li, “Energyefficient multi-uav-enabled multiaccess edge computing incorporating noma,” IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5613–5627, 2020.

[11] Mirzaei, A. (2022). A novel approach to QoS-aware resource allocation in NOMA cellular HetNets using multi-layer optimization. Concurrency and Computation: Practice and Experience, 34(21), e7068.

[12] T. Zhang, Y. Xu, J. Loo, D. Yang, L. Xiao, and Y. Zhao, “Joint computation and communication design for uav-assisted mobile edge computing in iot,” IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5505–5516, 2020.

[13] Z. Liu, X. Tan, M. Wen, S. Wang, C. Liang, and Q. Zhao, “An energyefficient selection mechanism of relay and edge computing in uavassisted cellular networks,” IEEE Transactions on Green Communications and Networking, vol. 5, no. 3, pp. 1306–1318, 2021.

[14] Mohajer, Amin, Javad Hajipour, and Victor CM Leung. "Dynamic Offloading in Mobile Edge Computing with Traffic-Aware Network Slicing and Adaptive TD3 Strategy." IEEE Communications Letters (2024).

[15] Yang, Jiuting, and Amin Mohajer. "Multi objective constellation optimization and dynamic link utilization for sustainable information delivery using PD-NOMA deep reinforcement learning." Wireless Networks (2024): 1-21.

[16] Somarin, A. M., Barari, M., & Zarrabi, H. (2018). Big data based self-optimization networking in next generation mobile networks. Wireless Personal Communications, 101(3), 1499-1518.

[17] Kuang, Shuhong, Jiyong Zhang, and Amin Mohajer. "Reliable information delivery and dynamic link utilization in MANET cloud using deep reinforcement learning." Transactions on Emerging Telecommunications Technologies 35, no. 9 (2024): e5028.

[18] Hua, Yuxiu, Rongpeng Li, Zhifeng Zhao, Xianfu Chen, and Honggang Zhang. "GAN-powered deep distributional reinforcement learning for resource management in network slicing." IEEE Journal on Selected Areas in Communications 38, no. 2 (2019): 334-349.

[19] X. Qin, Z. Song, Y. Hao, and X. Sun, "Joint Resource allocation and trajectory optimization for multi-uav-assisted multi-access mobile edge computing," IEEE Wireless Communications Letters, vol. 10, no. 7, pp. 1400–1404, 2021.

[20] Wang, Qianxing, Wei Li, and Amin Mohajer. "Load-aware continuous-time optimization for multi-agent systems: Toward dynamic resource allocation and real-time adaptability." Computer Networks 250 (2024): 110526.

[21] H. Hu, Z. Chen, F. Zhou, Z. Han, and H. Zhu, "Joint Resource and trajectory optimization for heterogeneous-uavs enabled aerial-ground cooperative computing networks," IEEE Transactions on Vehicular Technology, vol. 72, no. 6, pp. 7119–7133, 2023.

[22] Mirzaei, A., Barari, M., & Zarrabi, H. (2019). Efficient resource management for non-orthogonal multiple access: A novel approach towards green hetnets. Intelligent Data Analysis, 23(2), 425-447.

[23] Gu, LiFen, and Amin Mohajer. "Joint throughput maximization, interference cancellation, and power efficiency for multi-IRS-empowered UAV communications." Signal, Image and Video Processing 18, no. 5 (2024): 4029-4043.

[24] G. Chen, Q. Wu, R. Liu, J. Wu, and C. Fang, "Irs aided mec systems with binary offloading: A unified framework for dynamic irs beamforming,"IEEE Journal on Selected Areas in Communications, vol. 41, no. 2, pp. 349–365, 2023.

[25] X. Li, Y. Qin, J. Huo, and W. Huangfu, "Computation offloading and trajectory planning of multi-uav-enabled mec: A knowledge-assisted multiagent reinforcement learning approach, IEEE Internet of Things Journal, 2023.

[26] Yang, Ting, Jiabao Sun, and Amin Mohajer. "Queue stability and dynamic throughput maximization in multi-agent heterogeneous wireless networks." Wireless Networks (2024): 1-27.

[27] Mirzaei, A., & Rahimi, A. (2019). A Novel Approach for Cluster Self-Optimization Using Big Data Analytics. Information Systems & Telecommunication, 50.

[28] Y. Gu, C. Yin, Y. Guo, B. Xia, and Z. Chen, "Communicationcomputation- aware user association in mec hetnets: A meta-analysis," IEEE Transactions on Wireless Communications, vol. 22, no. 9, pp. 6090–6105, 2023.

[29] Zhang, Qi, Zhigang Li, Zhenteng Qin, Xiaochuan Sun, and Haijun Zhang. "Temporal Feature-Enhanced Deep Reinforcement Learning for RAN Slicing with User Mobility." IEEE Communications Letters (2023).

[30] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems," IEEE Journal on Selected Areas in Communications, vol. 36, no. 9, pp. 1927–1941, 2018.

[31] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for uav-enabled mobile edge computing systems,"IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1879–1892, 2019.

[32] Zhao, Zhongyong, Yu Chen, Jiangnan Liu, Yingying Cheng, Chao Tang, and Chenguo Yao. "Evaluation of operating state for smart electricity meters based on transformer–encoder–BiLSTM." IEEE Transactions on Industrial Informatics 19, no. 3 (2022): 2409-2420.

[33] Mohajer, Amin, Maryam Bavaghar, Rashin Saboor, and Ali Payandeh. "Secure dominating set-based routing protocol in MANET: Using reputation." In 2013 10th International ISC Conference on Information Security and Cryptology (ISCISC), pp. 1-7. IEEE, 2013.

[34] Y. Xu, T. Zhang, Y. Liu, D. Yang, L. Xiao, and M. Tao, "Cellular connected multi-uav mec networks: An online stochastic optimization approach," IEEE Transactions on Communications, vol. 70, no. 10, pp. 6630–6647, 2022.

[35] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2024). Metaheuristic and Data Mining Algorithms-based Feature Selection Approach for Anomaly Detection. IETE Journal of Research, 1-15.

[36] Li, Rongpeng, Chujie Wang, Zhifeng Zhao, Rongbin Guo, and Honggang Zhang. "The LSTM-based advantage actor-critic learning for resource management in network slicing with user mobility." IEEE Communications Letters 24, no. 9 (2020): 2005-2009.

[37] L. Zhang, J. Li, Y. Wang, Z. Chen, Q. Liu, and Y. Sun, "Task offloading and trajectory control for uav-assisted mobile edge computing using deep reinforcement learning," IEEE Access, vol. 9, pp. 53 708–53 719, 2021.

[38] X. Zhang, J. Zhang, J. Xiong, L. Zhou, J. Wei, and H. Li, "Energy efficient multi-uav-enabled multiaccess edge computing incorporating noma," IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5613–5627, 2020.

[39] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multiagent deep reinforcement learning-based trajectory planning for multiuav assisted mobile edge computing," IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 1, pp. 73–84, 2021.

[40] T. Zhang, Y. Xu, J. Loo, D. Yang, L. Xiao, and Y. Zhao, "Joint computation and communication design for uav-assisted mobile edge computing in iot," IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5505–5516, 2020.

[41] Z. Liu, X. Tan, M. Wen, S. Wang, C. Liang, and Q. Zhao, "An energy efficient selection mechanism of relay and edge computing in uavassisted cellular networks," IEEE Transactions on Green Communications and Networking, vol. 5, no. 3, pp. 1306–1318, 2021.

[42] Yan, Dandan, Benjamin K. Ng, Wei Ke, and Chan-Tong Lam. "Deep reinforcement learning based resource allocation for network slicing with massive MIMO." IEEE Access (2023).

[43] C.-Y. Hsieh, Y. Ren, and J.-C. Chen, "Edge-cloud offloading: Knapsack potential game in 5g multi-access edge computing," IEEE Transactions on Wireless Communications, vol. 22, no. 4, pp. 3124–3136, 2023.

[44] N. Zhao, C. Xu, W. Zhang, S. Yang, G.-M. Muntean, and F. Zhou,"5g-enabled uav-to community offloading: Joint trajectory design and task scheduling," IEEE Journal on Selected Areas in Communications, vol. 39, no. 11, pp. 3306–3320, 2021.

[45] H. Guo and J. Liu, "Uav-enhanced intelligent offloading for internet of things at the edge, IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 2737–2746, 2020.

[46] Wang, Zhaoying, Yifei Wei, F. Richard Yu, and Zhu Han. "Utility optimization for resource allocation in multi-access edge network slicing: A twin-actor deep deterministic policy gradient approach." IEEE Transactions on Wireless Communications 21, no. 8 (2022): 5842-5856.

[47] X. Qin, Z. Song, Y. Hao, and X. Sun, "Joint Resource allocation and trajectory optimization for multi-uav-assisted multi-access mobile edge computing," IEEE Wireless Communications Letters, vol. 10, no. 7, pp. 1400–1404, 2021.

[48] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energyefficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization," IEEE Transactions on Vehicular Technology, vol. 69, no. 3, pp. 3424–3438, 2020.

[49] Wang, Yue, Yu Gu, and Xiaofeng Tao. "Edge network slicing with statistical QoS provisioning." IEEE Wireless Communications Letters 8, no. 5 (2019): 1464-1467.

[50] H. Guo and J. Liu, "Uav-enhanced intelligent offloading for internet of things at the edge, IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 2737–2746, 2020.

Journal of Optimization of Soft Computing (JOSC)

Vol. 2, Issue 4, pp: (16-20), Winter-2024
Journal homepage: https://sanad.iau.ir/journal/josc

Paper Type (Research paper)

A Review of Feature Selection

Jafar Abdollahi[1], Babak Nouri-Moghaddam[1], Naser Mikaeilvand[2], Sajjad Jahanbakhsh Gudakahriz[3], Ailin Khosravani[1], Abbas Mirzaei[1]

1. Department of Computer Engineering, Ardabil Branch, Islamic Azad University, Ardabil, Iran
2. Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran
3. Department of Computer Engineering, Germi Branch, Islamic Azad University, Germi, Iran

Article Info

Article History:
Received: 2024/11/28
Revised: 2024/12/29
Accepted: 2025/03/13

DOI:

Keywords:
Data Mining; Medical Applications; Dimension Reduction; Feature Selection

**Corresponding Author's Email Address:*

Abstract

Feature selection is a preprocessing technique that identifies the salient features of a given scenario. It has been used in the past for a wide range of problems, including intrusion detection systems, financial problems, and the analysis of biological data. Feature selection has been especially useful in medical applications, where it may help identify the underlying reasons for an illness in addition to reducing dimensionality. We provide some basic concepts of medical applications and the necessary background information on feature selection. We review the most recent feature selection methods developed for and applied to medical problems, covering a broad spectrum of applications including medical imaging, DNA microarray data analysis, and biomedical signal processing. A case study of two medical applications utilizing actual patient data is used to demonstrate the usefulness of applying feature selection techniques to medical challenges and to highlight how these methods function in practical scenarios.

1. Introduction

Feature selection is one way to reduce dimensionality; in this strategy, only significant traits are retained while superfluous and redundant ones are discarded. Two ways that a reduction in input dimensionality might improve performance are either decreasing learning time and model complexity or increasing generalization capabilities and classification accuracy. Using the right features might improve problem understanding and reduce measurement expenses. In certain situations, the impact of feature selection may be substantial; for example, in microarray data analysis, just two of the 7129 features may be used to improve classification performance [1].

There are two kinds of feature selection models:

- Supervised Models: The technique that selects features based on the output label class is known as supervised feature selection.
- Unsupervised Models: An approach that selects features without requiring knowledge of the output label class is known as unsupervised feature selection.

In many applications, it has been necessary to combine pattern recognition algorithms with FS techniques, since many of them were not designed to handle large amounts of irrelevant data at first. Preventing overfitting and enhancing model performance—more specifically, prediction performance in supervised classification and improved cluster detection in clustering—are the primary objectives of feature selection. Other objectives include (a) producing faster and more efficient models and (b) gaining a deeper comprehension of the underlying processes that generated the data. Nevertheless, the advantages of feature selection strategies are not without a price, as the search for a subset of pertinent characteristics raises the bar for modeling complexity. We must now determine the model's optimal parameters for the optimal feature subset in addition to optimizing its parameters for the full feature subset, since there is no guarantee that the model's ideal parameters for the entire feature set will also be optimal for the optimal feature subset. Thus, identifying the optimal subset of pertinent attributes expands the scope of the search within the model hypothesis space. Every feature selection technique uses a different technique to include this search in the extra space of feature subsets when choosing a model [2, 4, 5].

Filter approaches assess the significance of the features by concentrating on the intrinsic properties of the data. Generally, features are ranked according to their relevance, and those with lower scores are ignored. This selection of attributes is then given as input to the classification algorithm. Because of the advantages of filter approaches—which include their simplicity and speed in computation, their independence from the classification algorithm, and their ability to scale to extremely high-dimensional datasets—only one feature selection process is needed before multiple classifiers can be evaluated [2,].

Unlike filter techniques, which tackle the problem of finding a suitable feature subset independently of the model selection phase, wrapper approaches incorporate the model hypothesis search into the feature subset search. In this scenario, various feature subsets are generated and evaluated in the space of possible feature subsets utilizing a predefined search method [2].

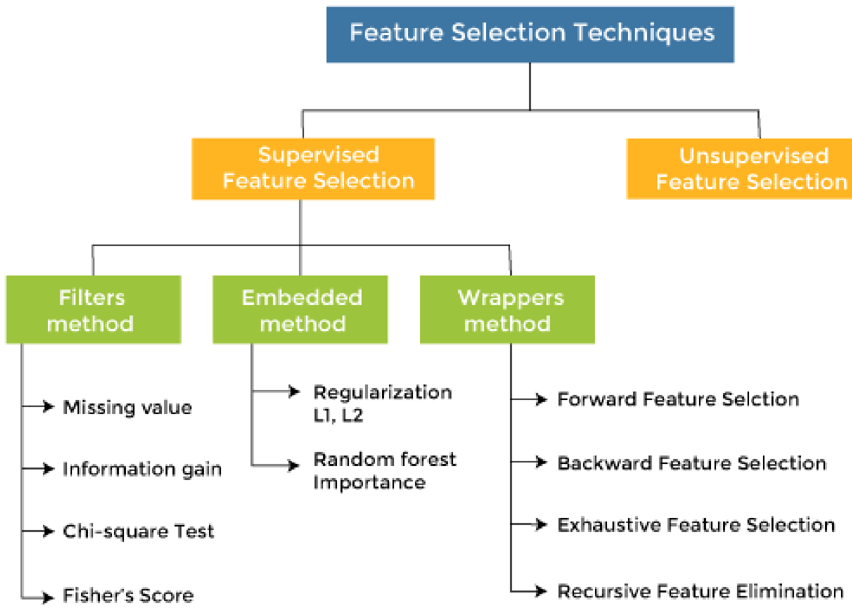


Figure 1: Overview of Feature Selection Strategies [2]

2. Choosing Feature

Medical image and healthcare analysis, including diabetes [15, 24, 32], breast cancer [16, 25], healthcare system [17], forecasting [18], stock market [19], stroke [20], COVID-19 [21], types of epidemic [22], medicinal plants [23], heart [26], lung cancer [27, 30], social networks [28], prediction of diphenhydramine [29], and bupro, have all benefited from the successful application of artificial intelligence, which includes machine learning and deep learning. We present a comprehensive review of feature selection methods applied in medicine over the past five years, some developed on the fly to tackle specific problems. Specifically, feature selection has been applied in three main medical fields: biomedical signal processing, DNA microarray data, and medical imaging. We then go on to discuss current advancements in each of these fields. Then, we discuss how feature selection is applied to two actual medical image analysis situations and show the benefits that follow from doing so [1,39, 44]. The following is a summary of feature choices.

- Feature Selection: Select a subset of input features from the dataset.
- Unsupervised: Do not use the target variable (e.g. remove redundant variables).
- Correlation
- Supervised: Use the target variable (e.g. remove irrelevant variables).
- Wrapper: Search for well-performing subsets of features.
- RFE
- Filter: Select subsets of features based on their relationship with the target.
- Statistical Methods
- Feature Importance Methods

- Intrinsic: Algorithms that conduct automated feature selection during training.
- Decision Trees
- Dimensionality Reduction: Project input data into a lower-dimensional feature space.

The figure above offers an overview of the hierarchy of feature selection strategies.

A. Primary Concepts

In line with how they combine the selection algorithm and the model development, the feature selection strategies are frequently categorized into three forms.

B. Filter Method

Filtering strategy for picking features: Methods of the filter type pick variables without attention to the model. They are just reliant on universal qualities like the correlation with the expected variable. Filtering strategies reduce the least fascinating aspects. The following variables will be added to a regression model or classification scheme used to classify or forecast data. These approaches offer good computational efficiency and are resistant to overfitting. When filter algorithms do not take into consideration the relationships between variables, duplicated variables are typically picked. However, more complicated features, like the Fast Correlation Based Filter (FCBF) algorithm, aim to decrease this problem by removing variables that are highly linked with one another.

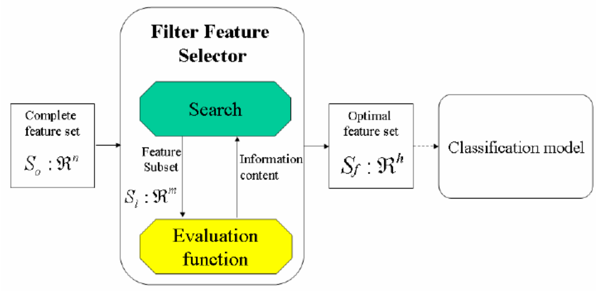


Figure 2: The Hierarchy of Feature Selection Methods [29]

C. Wrapper Method

Wrapper approach for feature selection: Wrapper techniques, in contrast to filter operations, look at subsets of variables, which makes it possible to find any possible interactions between variables. Overfitting becomes more likely when there are insufficient observations, and computation time grows dramatically as there are more variables [41, 42, 43].

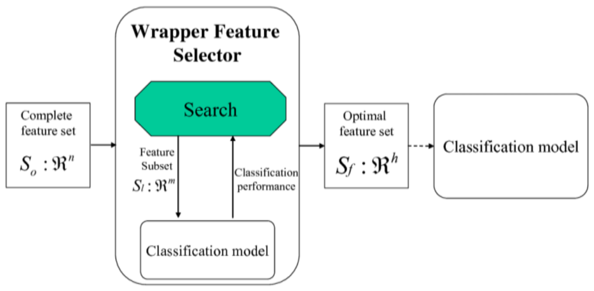


Figure 2: Wrapper Method in Feature Selection [43]

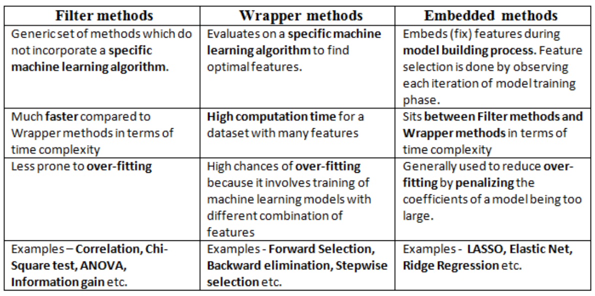
D. Embedded Methodology

Choosing features using the embedded method: Embedded strategies have recently been created with the goal of combining the advantages of the two previous approaches. A learning algorithm, such as the FRMT technique, uses its own variable selection mechanism to carry out feature selection and classification simultaneously [40, 45].

3. Finding

In many bioinformatics applications, feature selection algorithms are needed. Dedicated bioinformatics applications have yielded a wide range of recently proposed methods to supplement the large body of previously developed methods in the fields of data mining and machine learning [2, 46]. In recent years, there has been a notable increase in the use of feature selection approaches in medical datasets. The challenging task in feature selection is to identify the perfect subset of relevant and non-redundant qualities that will provide an optimal solution without adding to the complexity of the modeling process. Therefore, it's critical to draw attention to recent advancements in this area and educate practitioners on feature selection strategies that have worked well with medical data sets. The findings demonstrate that most feature selection methods now in use are based on univariate ranking, which overlooks the stability of the selection algorithms, interactions between variables, and the requirement for additional features to attain very high accuracy. Less attributes may still lead to maximum classification accuracy, but more work has to be done in this area [3, 14, 33-38, 47].

Tables1: Summary of Feature Selection Methods [47]



Filter methods	**Wrapper methods**	**Embedded methods**
Generic set of methods which do not incorporate a **specific machine learning algorithm.**	Evaluates on a **specific machine learning algorithm** to find optimal features.	Embeds (fix) features during **model building process.** Feature selection is done by observing each iteration of model training phase.
Much **faster** compared to Wrapper methods in terms of time complexity	**High computation time** for a dataset with many features	Sits **between Filter methods and Wrapper methods** in terms of time complexity
Less prone to **over-fitting**	High chances of **over-fitting** because it involves training of machine learning models with different combination of features	Generally used to reduce **over-fitting** by **penalizing** the coefficients of a model being too large.
Examples – **Correlation, Chi-Square test, ANOVA, Information gain** etc.	Examples - **Forward Selection, Backward elimination, Stepwise selection** etc.	Examples - **LASSO, Elastic Net, Ridge Regression** etc.

4. Conclusion

Feature selection is a fundamental technique for enhancing machine learning models by reducing dimensionality, improving accuracy, and optimizing computational efficiency. This review has highlighted the significance of feature selection in various medical applications, including biomedical signal processing, DNA microarray analysis, and medical imaging. While existing methods provide effective solutions for reducing irrelevant and redundant features, many challenges remain in achieving an optimal subset of features that balances performance and computational cost. Recent advances have shown that hybrid models combining filter, wrapper, and embedded approaches yield promising results. However, issues such as model stability, feature interaction, and scalability need further exploration. Future research should focus on developing more robust feature selection techniques tailored for complex medical datasets, ensuring better diagnostic accuracy and predictive performance in healthcare applications.

References

[1] Remeseiro, B., & Bolon-Canedo, V. (2019). A review of feature selection methods in medical applications. Computers in biology and medicine, 112, 103375.

[2] Saeys, Y., Inza, I., & Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. bioinformatics, 23(19), 2507-2517.

[3] Mwadulo, M. W. (2016). A review on feature selection methods for classification tasks.

[4] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. ACM computing surveys (CSUR), 50(6), 1-45.

[5] Duan, H., & Mirzaei, A. (2023). Adaptive Rate Maximization and Hierarchical Resource Management for Underlay Spectrum Sharing NOMA HetNets with Hybrid Power Supplies. Mobile Networks and Applications, 1-17.

[6] Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. Computers & Electrical Engineering, 40(1), 16-28.

[7] Dash, M., & Liu, H. (1997). Feature selection for classification. Intelligent data analysis, 1(1-4), 131-156.

[8] Somarin, A. M., Barari, M., & Zarrabi, H. (2018). Big data based self-optimization networking in next generation mobile networks. Wireless Personal Communications, 101(3), 1499-1518.

[9] Liu, H., & Motoda, H. (Eds.). (2007). Computational methods of feature selection. CRC press.

[10] Koller, D., & Sahami, M. (1996, July). Toward optimal feature selection. In ICML (Vol. 96, No. 28, p. 292).

[11] Venkatesh, B., & Anuradha, J. (2019). A review of feature selection and its methods. Cybernetics and information technologies, 19(1), 3-26.

[12] Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. Neurocomputing, 300, 70-79.

[13] Zhao, Z., Morstatter, F., Sharma, S., Alelyani, S., Anand, A., & Liu, H. (2010). Advancing feature selection research. ASU feature selection repository, 1-28.

[14] Saeys, Y., Abeel, T., & Van de Peer, Y. (2008). Robust feature selection using ensemble feature selection techniques. In Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II 19 (pp. 313-325). Springer Berlin Heidelberg.

[15] Abdollahi, J., Moghaddam, B. N., & Parvar, M. E. (2019). Improving diabetes diagnosis in smart health using genetic-based Ensemble learning algorithm. Approach to IoT Infrastructure. Future Gen Distrib Systems J, 1, 23-30.

[16] Abdollahi, J., Keshandehghan, A., Gardaneh, M., Panahi, Y., & Gardaneh, M. (2020). Accurate detection of breast cancer metastasis using a hybrid model of artificial intelligence algorithm. Archives of Breast Cancer, 22-28.

[17] Nematollahi, M., Ghaffari, A., & Mirzaei, A. (2024). Task and resource allocation in the internet of things based on an improved version of the moth-flame optimization algorithm. Cluster Computing, 27(2), 1775-1797.

[18] Abdollahi, J., Irani, A. J., & Nouri-Moghaddam, B. (2021). Modeling and forecasting Spread of COVID-19 epidemic in Iran until Sep 22, 2021, based on deep learning. arXiv preprint arXiv:2103.08178.

[19] Abdollahi, J., & Mahmoudi, L. Investigation of artificial intelligence in stock market prediction studies. In 10th International Conference on Innovation and Research in Engineering Science.

[20] Narimani, Y., Zeinali, E., & Mirzaei, A. (2022). QoS-aware resource allocation and fault tolerant operation in hybrid SDN using stochastic network calculus. Physical Communication, 53, 101709.

[21] Abdollahi, J. (2020). A review of Deep learning methods in the study, prediction and

management of COVID-19. In 10th International Conference on Innovation and Research in Engineering Science.

[22] Abdollahi, J., & Mahmoudi, L. (2022, February). An Artificial Intelligence System for Detecting the Types of the Epidemic from X-rays: Artificial Intelligence System for Detecting the Types of the Epidemic from X-rays. In 2022 27th International Computer Conference, Computer Society of Iran (CSICC) (pp. 1-6). IEEE.

[23] Abdollahi, J. (2022, February). Identification of medicinal plants in ardabil using deep learning: identification of medicinal plants using deep learning. In 2022 27th International Computer Conference, Computer Society of Iran (CSICC) (pp. 1-6). IEEE.

[24] Abdollahi, J., & Nouri-Moghaddam, B. (2022). Hybrid stacked ensemble combined with genetic algorithms for diabetes prediction. Iran Journal of Computer Science, 5(3), 205-220.

[25] Abdollahi, J., Davari, N., Panahi, Y., & Gardaneh, M. (2022). Detection of Metastatic Breast Cancer from Whole-Slide Pathology Images Using an Ensemble Deep-Learning Method: Detection of Breast Cancer using Deep-Learning. Archives of Breast Cancer, 364-376.

[26] Jahanbakhsh Gudakahriz, S., Momtaz, V., Nouri-Moghadam, B., Mirzaei, A., & Vajed Khiavi, M. (2025). Link life time and energy-aware stable routing for MANETs. International Journal of Nonlinear Analysis and Applications.

[27] Javadzadeh Barzaki, M. A., Negaresh, M., Abdollahi, J., Mohammadi, M., Ghobadi, H., Mohammadzadeh, B., & Amani, F. (2022, July). USING DEEP LEARNING NETWORKS FOR CLASSIFICATION OF LUNG CANCER NODULES IN CT IMAGES. In Iranian Congress of Radiology (Vol. 37, No. 2, pp. 34-34). Iranian Society of Radiology.

[28] Khavandi, H., Moghadam, B. N., Abdollahi, J., & Branch, A. (2023). Maximizing the Impact on Social Networks using the Combination of PSO and GA Algorithms. Future Generation in Distributed Systems, 5, 1-13.

[28] Mehrpour, O., Saeedi, F., Abdollahi, J., Amirabadizadeh, A., & Goss, F. (2023). The value of machine learning for prognosis prediction of diphenhydramine exposure: National analysis of 50,000 patients in the United States. Journal of Research in Medical Sciences, 28(1), 49.

[29] Rad, K. J., & Mirzaei, A. (2022). Hierarchical capacity management and load balancing for HetNets using multi-layer optimisation methods. International Journal of Ad Hoc and Ubiquitous Computing, 41(1), 44-57.

[30] Mehrpour, O., Saeedi, F., Vohra, V., Abdollahi, J., Shirazi, F. M., & Goss, F. (2023). The role of decision tree and machine learning models for outcome prediction of bupropion exposure: A nationwide analysis of more than 14 000 patients in the United States. Basic & Clinical Pharmacology & Toxicology, 133(1), 98-110.

[31] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2024). The impact of financial ratio reduction on supervised methods' ability to detect financial statement fraud. Karafan Quarterly Scientific Journal.

[32] Tajidini, F., & Kheiri, M. J. (2023). Recent advancement in Disease Diagnostic using machine learning: Systematic survey of decades, comparisons, and challenges. arXiv preprint arXiv:2308.01319.

[33] Zargar, H. H., Zargar, S. H., Mehri, R., & Tajidini, F. (2023). Using VGG16 Algorithms for classification of lung cancer in CT scans Image. arXiv preprint arXiv:2305.18367.

[34] Mirzaei, A., Barari, M., & Zarrabi, H. (2019). Efficient resource management for non-orthogonal multiple access: A novel approach towards green hetnets. Intelligent Data Analysis, 23(2), 425-447.

[35] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2023). Financial Ratios and Efficient Classification Algorithms for Fraud Risk Detection in Financial Statements. International Journal of Industrial Mathematics.

[36] Tajidini, F., & Mehri, R. A survey of using Deep learning algorithms for the Covid-19 (SARS-CoV-2) pandemic: A review.

[37] Tajidini, F., & Piri, M. Machine Learning Methods for prediction of Diabetes: A Narrative.

[38] Mirzaei, A., & Najafi Souha, A. (2021). Towards optimal configuration in MEC Neural networks: deep learning-based optimal resource allocation. Wireless Personal Communications, 121(1), 221-243.

[39] Nematollahi, M., Ghaffari, A., & Mirzaei, A. (2024). Task offloading in Internet of Things based on the improved multi-objective aquila optimizer. Signal, Image and Video Processing, 18(1), 545-552.

[40] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2024). Metaheuristic and Data Mining Algorithms-based Feature Selection Approach for Anomaly Detection. IETE Journal of Research, 1-15.

[41] Nematia, Z., Mohammadia, A., Bayata, A., & Mirzaeib, A. (2024). Predicting fraud in financial statements using supervised methods: An analytical comparison. International Journal of Nonlinear Analysis and Applications, 15(8), 259-272.

[42] Jahandideh, Y., & Mirzaei, A. (2021). Allocating Duplicate Copies for IoT Data in Cloud Computing based on Harmony Search Algorithm. IETE Journal of Research, 1-14.

[43] Mikaeilvand, N., Ojaroudi, M., & Ghadimi, N. (2015). Band-Notched Small Slot Antenna Based on Time-Domain Reflectometry Modeling for UWB Applications. The Applied Computational Electromagnetics Society Journal (ACES), 682-687.

[44] Mikaeilvand, N. (2011). On solvability of fuzzy system of linear matrix equations. J Appl Sci Res, 7(2), 141-153.

[45] Allahviranloo, T., & Mikaeilvand, N. (2011). Non zero solutions of the fully fuzzy linear systems. Appl. Comput. Math, 10(2), 271-282.

[46] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2024). Fraud Risk Prediction in Financial Statements through Comparative Analysis of Genetic Algorithm, Grey Wolf Optimization, and Particle Swarm Optimization. Iranian Journal of Finance, 8(1), 98-130.

Journal of Optimization of Soft Computing (JOSC)

Vol. 2, Issue 4, pp: (21-27), Winter-2024
Journal homepage: https://sanad.iau.ir/journal/josc

Paper Type (Research paper)

Optimal Shape Investigation of Masonry Arch Bridges under Dynamic Loads Using Support Vector Machine

Kaveh Kumarci[1]

1. ***College of skills and entrepreneurship, Shahrekord branch, Islamic Azad University, Shahrekord, Iran***

Article Info

Article History:
Received: 2024/12/05
Revised: 2025/02/13
Accepted: 2025/02/15

DOI:

Keywords:
Masonry Arches, Dynamic Loads, Support Vector Machine (SVM), Structural Risk Minimization (SRM), Shape Optimization

**Corresponding Author's Email Address:kumarci_kaveh@yahoo.com*

Abstract

The objective of this study is to determine the optimal shape of masonry arches under dynamic loads using the Support Vector Machine (SVM) technique. This approach utilizes the principles of Structural Risk Minimization (SRM), which demonstrate superior performance compared to methods based on Empirical Risk Minimization (ERM). The research particularly focuses on the types of arches commonly used in traditional structures and their significance in ensuring structural stability and performance. The modeling, dynamic analysis, and shape optimization of a semi-circular arch are comprehensively explained using ANSYS 11 software and the SVM method. The necessity of this study lies in the critical role that the optimal shape of arches plays in enhancing the resilience and reducing the vulnerability of masonry structures against dynamic loads, especially given their widespread application in both historical and modern constructions. The main innovation of this research is the application of the Support Vector Machine as an advanced and less commonly employed method for arch shape optimization. For the first time, SRM principles are integrated with dynamic modeling and computational analysis, offering a novel framework for optimizing traditional structures.

1. Introduction

Masonry arch bridges have been integral components of architectural and engineering heritage for centuries, known for their aesthetic appeal and structural efficiency. These structures, prevalent in both historical and modern applications, require meticulous analysis to ensure their resilience, particularly under dynamic loads such as seismic activity and vehicular traffic. A critical aspect of their performance lies in the optimization of their geometric shape, which significantly influences their ability to withstand dynamic forces while maintaining stability and durability [1].

Dynamic analysis is a comprehensive time-history analytical method that evaluates the responses of structures to time-dependent excitations, such as earthquakes. By numerically integrating the equations of motion, this method provides a detailed understanding of time-varying displacements, strains, stresses, and forces within a structure. Such insights are essential for predicting the behavior of masonry arches under dynamic loads, enabling engineers to design and optimize structures that meet safety and performance requirements [2].

Previous research has explored various aspects of modeling, dynamic analysis, and shape optimization of masonry arches. These studies have demonstrated the significance of employing advanced computational tools like ANSYS software for conducting dynamic analyses. However, these methods are often computationally intensive, requiring significant time and resources to achieve accurate results. The reliance on traditional optimization techniques, primarily based on Empirical Risk Minimization (ERM), has further limited the efficiency and applicability of these approaches [3].

Despite the progress made, a notable gap exists in the integration of advanced machine learning techniques, such as Support Vector Machines (SVM), into the dynamic analysis and optimization of masonry arches. Traditional methods have struggled to balance computational efficiency with the precision required for analyzing complex structural behaviors. Furthermore, existing studies have not fully leveraged the principles of Structural Risk Minimization (SRM), which offer a more robust framework for predictive modeling compared to ERM-based techniques [4].

To address these limitations, the present study introduces an innovative framework that combines SVM with SRM principles for the dynamic analysis and shape optimization of masonry arches. By employing this approach, the computational burden of dynamic analysis is significantly reduced while maintaining high accuracy in results. Additionally, the integration of SVM into the optimization process represents a novel application in the field, filling a critical void in the current body of knowledge. This research not only advances the methodological tools available for structural optimization but also sets a precedent for future studies aiming to enhance the resilience and performance of masonry arch bridges under dynamic loads.

2. Literature review

This section discusses related research on Masonry Arch Bridges under Dynamic Loads**.** In [5], Authors developed a hybrid optimization framework combining genetic algorithms with finite element analysis to investigate the optimal shapes of masonry arches. Although this approach demonstrated improvements in optimization outcomes, it faced challenges in handling high-dimensional design spaces efficiently. Our SVM-based methodology addresses this limitation by offering robust performance in high-dimensional settings and ensuring scalability.Another noteworthy contribution by [6] utilized deep learning models to predict the dynamic stability of semicircular masonry arches. While their neural network models achieved high accuracy, the need for extensive training data and the risk of overfitting limited the practical application of their approach. Our method overcomes these issues by leveraging SVM, which requires smaller datasets and inherently avoids overfitting through SRM principles.In [7], the influence of material properties on the seismic performance of masonry arches was investigated using parametric analyses. Although the research provided a detailed understanding of material behavior, it lacked a systematic framework for shape optimization. Our research extends beyond material analysis to include comprehensive shape optimization, enhancing the overall resilience of masonry arches. Finally, in [8] authors examined the impact of geometric irregularities on the dynamic performance of masonry arches through numerical simulations. While the study highlighted critical geometric factors affecting stability, it did not incorporate advanced optimization methodologies. Our work fills this gap by integrating machine learning techniques directly into the optimization process, providing a more efficient and effective framework for analyzing and improving structural performance.

Despite the progress made, a notable gap exists in the integration of advanced machine learning techniques, such as Support Vector Machines (SVM), into the dynamic analysis and optimization of masonry arches. Traditional methods have struggled to balance computational efficiency with the precision required for analyzing complex structural behaviors. Furthermore, existing studies have not fully leveraged the principles of Structural Risk Minimization (SRM), which offer a more robust framework for predictive modeling compared to ERM-based techniques.To address these limitations, the present study introduces an innovative framework that combines SVM with SRM principles for the dynamic analysis and shape optimization of masonry arches. By employing this approach, the computational burden of dynamic analysis is significantly reduced while maintaining high accuracy in results. Additionally, the integration of SVM into the optimization process represents a novel application in the field, filling a critical void in the current body of knowledge. This research not only advances the methodological tools available for structural optimization but also sets a precedent for future studies aiming to enhance the resilience and performance of masonry arch bridges under dynamic loads.

3. Modeling, Analysis, and Shape Optimization of Arches Using ANSYS 11

Considering that in the optimization section, design variables, namely the thickness of the base and the thickness of the crown, need to be defined as parameters, key points in the modeling of the arch must be defined as follows[9].

4. Geometrical Modeling

For clarity, the semi-circular arch with the definition of key points as parameters is presented in (Figure 1), where the coordinates of the key points are defined as follows (Table1):

Table 1: Coordinates of Key Points of the Semi-Circular Arch

point	1	2	3	4	5	6	7
X coordinates	0	R	-R	0	R+t0	-(R+t0)	0
Y coordinates	0	0	0	R	0	0	R+t1

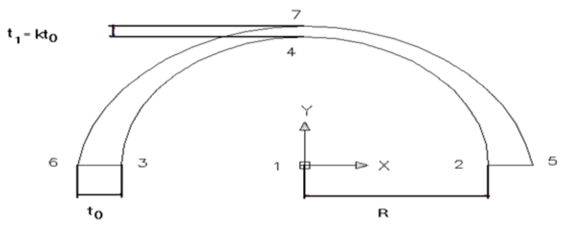


Figure 1: Semi-Circular Arch [8]

Modeling the arch in this way means that the gradual reduction in thickness from the base to the crown contributes to the stability of the arch. It is worth noting that in the modeled arch, the thickness decreases linearly from the base to the crown. Additionally, the thickness of the arch in the longitudinal direction is equal to 20 units. The displacements of the support nodes are set to zero, and the shear force is unable to displace them. Furthermore, the masonry consists of brick and mortar, considered as homogeneous materials with properties listed in Table 2, and the coefficients involved in the nonlinear and non-elastic analysis listed in Table 3 are taken into account.

Table 2: Characteristics of Masonry Materials

Density (ρ) kg/m^3	***1460*[6]**
Elastic Modulus (E) N/m^2	5×10^8 **[7]**
Allowable Tensile Stress (f_t) N/m^2	0.5×10^5 **[6,7,8]**
Poisson's Ratio (υ)	***0.17* [8]**

Table 3: Coefficients Influencing Nonlinear Non-Elastic Analysis

Shear Transfer Coefficient for Open Crack	0.1[6]
Shear Transfer Coefficient for closed Crack	0.9[7]
Allowable Tensile Stress N/m^2 (f_t)	5×10^4 [6, 7, 8]
Allowable compress Stress N/m^2 (f_c)	5×10^5 [6,7]

5. Support Vector Machine:

(SVM) is a machine learning method based on the statistical learning theory proposed by Vapnik and his colleagues in the 1990s. In SVM, the principles of Structural Risk Minimization (SRM) are employed, while other methods rely on Empirical Risk Minimization (ERM). It has been demonstrated that SRM principles perform better than ERM in terms of functionality. SVM is generally used for binary or multiclass classification and regression problems [10].

Like many other machine learning methods, SVM involves a model construction process consisting of two stages: training and testing. At the end of the training phase, the generalization capability of the trained model is evaluated using test data. In summary, the main operation of SVM in solving regression problems can be stated as follows:

1. Support Vector Machine approximates the regression function using a linear function.
2. Support Vector Machine performs regression operations with a function where the deviation from the actual value is less than ε (loss function).
3. By minimizing the structural risk, Support Vector Machine provides the best solution [11].

In methods such as artificial neural networks, empirical risk minimization principles are used to achieve the best solution. Minimizing empirical risk ensures the appropriate performance of the model on training data, but there is no guarantee of proper generalization. Therefore, in this method, proper network design is necessary to improve the generalization performance of the model. The goal of structural risk minimization is to optimize the

generalization capability of the model while minimizing empirical risk simultaneously [12].
Solving the regression problem in SVM involves approximating the regression function using a linear function f(x) =<w.x>+b. on a set containing a sample such as
{(x1,y1),….(x1,y1)ϵ Rn, yϵR} Translated academically, it becomes: to be able to estimate output values based on inputs. In the above equation, x is the input vector
(w,b) ϵ RN×R The controlling parameters of the function f are represented by <w.x>, indicating the inner product. For solving the regression problem, the Vapnik loss function is used, where a minimum error of ε can be ignored. This loss function is defined in equation (1) as follows:

$$L_\varepsilon(y) = |y - f(x)|_\varepsilon = \begin{cases} 0 & |y-f(x)|\le\varepsilon \\ |y - f(x)| - \varepsilon & \text{otherwise} \end{cases} \qquad (1)$$

Lε(y) represents the loss function and ε is the allowable error in the loss function. The controlling parameters of the optimal regression function are obtained by solving the following optimization problem:

$$\begin{aligned} & Minimise\,\Phi(W,\zeta^*,\zeta) = \frac{\|w\|^2}{2} + C\left(\sum \zeta_i^* + \sum \zeta\right) \\ & y_i - \left((W.X_i) + b\right) \le \varepsilon + \zeta_i \\ & Subject\ to\ \left((W.X_i) + b\right) - y_i \le \varepsilon + \zeta^*_i \\ & \zeta_i, \zeta_i^* \ge 0 \end{aligned} \qquad (2)$$

In the equation (2), ζ's are slack variables. These variables, along with the loss function, are depicted in Figure 2. To solve the optimization problem above, the Lagrange function is written according to equation (2) using the theory of Lagrange multipliers.

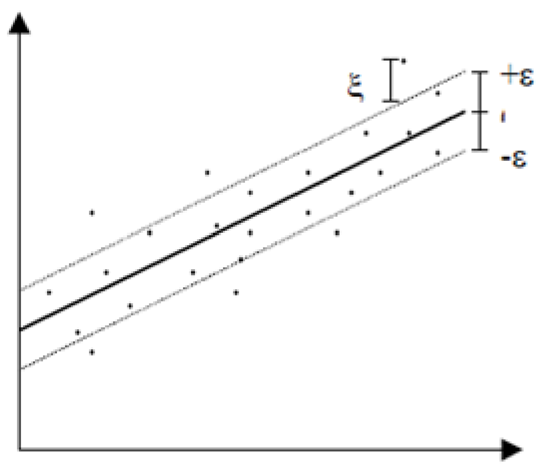


Figure 2: Vapnik's Loss Function and Slack Variables

With the maximization of the above function under the following constraints, the values of a and a* are obtained. These coefficients are referred to as Lagrange multipliers.
The optimization problem above can be solved using Quadratic Programming (QP) methods, thus achieving a definite global extremum. Consequently, the risk of overfitting these data points is higher. Therefore, support vectors do not lie within the margin band. Hence, controls the number of support vectors[13]. With the help of Lagrange multipliers and support vectors, the optimal response control parameters are calculated as follows:
In Equation 7, Xr and Xs are two support vectors.
For constructing a Support Vector Machine (SVM) model, the parameters C and are defined by the user. Parameter C is a regularization parameter and can take values from zero to infinity. Its role is to balance between minimizing empirical risk and maximizing generalization capability. Parameter can also take values from zero to infinity. Its value is crucial in the context of support vectors and consequently, the model's performance. Linear regression problem in SVM can be easily extended to non-linear regression. For this purpose, kernel functions are used [14]. Various kernels have been recognized so far, but the successful application of polynomial and radial basis function (rbf) kernels in geotechnical engineering problems has been reported. Thus, in the case of non-linear regression in SVM, the control parameters of the optimal function are calculated with the following equations:

6. Modeling arches using Support Vector Machines (SVM)

To generate and evaluate a Support Vector Machine (SVM)-based model for predicting the dynamic response of concrete arches under seismic force, 300 arch samples analyzed by ANSYS software are used. Each arch sample includes 3 independent variables: arch radius, base thickness, and crown thickness, and one dependent variable: maximum arch tensile stress. The range of these parameters in this study is defined as follows: arch radius (4 to 8 meters), base thickness (0.8 to 1.4 meters), and crown thickness (0.2 to 0.4 meters).
For creating the SVM model, the data are divided into two sets, training and evaluation, with a ratio of 70 to 30 (210 samples for training and 90 samples for evaluation). The desired model is generated using the training dataset, and its performance in predicting the desired population is evaluated using data not experienced during the model training (test dataset). Moreover, the radial

basis function (rbf) kernel, chosen as the best kernel function in various research studies, is used as the kernel function in this study[15]. To achieve a better model, multiple models are created by combining different combinations of kernel function parameters (C, and ζ), and their performance is evaluated. Additionally, the prediction results of the model are presented using statistical indices such as the correlation coefficient (R) and the root mean square error (RMSE). The correlation coefficient is a measure of the conformity of predicted values to measured values and is calculated according to the following equation.

Moreover, the value of RMSE, which is a measure for error estimation, is calculated according to the following equation.

Tables 4 to 6 present the results obtained from the generated models based on different combinations of parameters C, , and ζ.

Table 4: Model evaluation for various values of the kernel function parameter ζ

ζ	Train Set		Test Set	
	R	RMSE	R	RMSE
0.5	0.8324	0.2134	0.6914	0.1424
1	0.8873	0.2542	0.7105	0.1804
10	0.9132	0.0422	0.8123	0.1924
50	0.9732	0.1393	0.8012	0.0834
100	0.9023	0.2059	0.9145	0.1425
200	0.9802	0.1942	0.9014	0.1804
300	0.8931	0.1954	0.7204	0.1643
ε = .002 C=120				

Table 5: Model evaluation for various values of the kernel function parameter

ε	Train Set		Test Set	
	R	RMSE	R	RMSE
0.0001	0.8753	0.1246	0.7406	0.0245
0.001	0.8472	0.0754	0.8520	0.0810
0.005	0.9123	0.0864	0.9025	0.1149
0.01	0.7856	0.1825	0.8205	0.1820
0.05	0.7750	0.1342	0.7525	0.1025
0.1	0.8253	0.2305	0.8206	0.2150
ζ = 45 C=120				

Table 6: Model evaluation for various values of the kernel function parameter c

C	Train Set		Test Set	
	R	RMSE	R	RMSE
0.1	0.6892	0.2025	0.6027	0.1486
1	0.8402	0.1840	0.8242	0.1085
10	0.7920	0.1820	0.8295	0.0895
50	0.8154	0.1234	0.7930	0.0702
100	0.8682	0.0804	0.8206	0.1079
150	0.9104	0.0865	0.9253	0.0802
200	0.9425	0.9104	0.9874	0.9795
ε = .002 ζ =45				

7. Conclusion

The overall goal of this research is to utilize a nonlinear Support Vector Machine (SVM) model along with a radial basis function (rbf) kernel for predicting the dynamic response of concrete arches under seismic force. To this end, a dataset consisting of 300 arch samples analyzed by

ANSYS software is divided into a 70 to 30 ratio for training and evaluation datasets (Figure 3).

Finally, after determining the best SVM model, which exhibits adequate accuracy in predicting the dynamic responses of arches compared to actual results, the kernel function parameters (C, , and ζ) as well as the values of R (correlation coefficient) and RMSE (root mean square error) are presented as determinant parameters in selecting the best SVM model. Figure 4 compares the maximum tensile stress calculation by Support Vector Machine and ANSYS software. The results of the study indicate that the Support Vector Machine has an error ranging from 11 to 17 compared to the results obtained by ANSYS software.

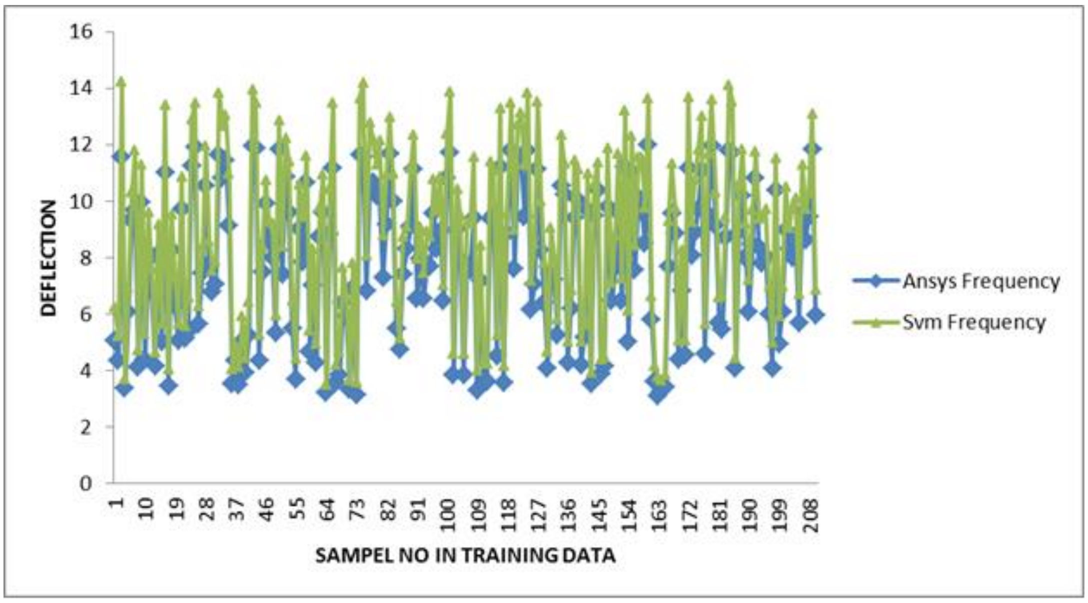


Figure 3: Comparison plot of maximum tensile stress calculated by ANSYS software and SVM

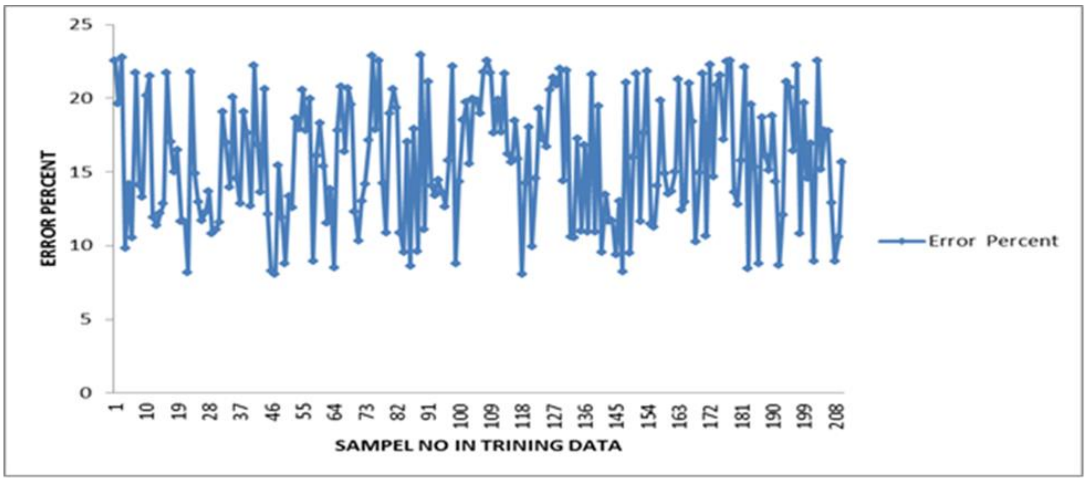


Figure 4: Percentage error plot of maximum tensile stress calculation by SVM software compared to ANSYS software

8. References

[1] Alpaslan, E., Hacıefendioğlu, K., Yılmaz, M. F., Demir, G., Mostofi, F., & Toğan, V. (2024). Structural Modal Calibration of Historical Masonry Arch Bridge by Using a Novel Deep Neural Network Approach. *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, *48*(1), 329-352.

[2] Cabanzo, C., Mendes, N., Akiyama, M., Lourenço, P. B., & Matos, J. C. (2025). Probabilistic framework for seismic performance assessment of a multi-span masonry arch bridge

employing surrogate modeling techniques. *Engineering Structures*, *325*, 119399.

[3] Azar, A. B., & Sari, A. (2024). Blast resistance of CFRP composite strengthened masonry arch bridge under close-range explosion. *Advances in Bridge Engineering*, *5*(1), 26.

[4] PANTO, B., Ortega, J., Grosman, S., Oliveira, D. V., Lourenço, P. B., Macorini, L., & Izzuddin, B. A. Advanced Calibration of a 3d Masonry Arch Bridge Model Using Non-Destructive Testing Information and Numerical Optimisation. *Available at SSRN 4732134*.

[5] Yuan, Y., Chen, H., Wang, J., Wang, W., & Chen, X. (2025). Additive manufacturing of catenary arch structure design: microstructure, mechanical properties and numerical simulation. *Journal of Materials Research and Technology*.

[6] Duan, J., Yan, H., Tao, C., Wang, X., Guan, S., & Zhang, Y. (2025). Integration of Finite Element Analysis and Machine Learning for Assessing the Spatial-Temporal Conditions of Reinforced Concrete. *Buildings*, *15*(3), 435.

[7] Colmenarez, J. A., Dong, P., Lee, J., Wilson, D. L., & Gu, L. (2025). Evaluating the Influence of Morphological Features on the Vulnerability of Lipid-Rich Plaques During Stenting. *Journal of Biomechanical Engineering*, *147*(2).

[8] Liu, B., Collier, J., & Sarhosis, V. (2025). Digital image correlation based crack monitoring on masonry arch bridges. *Engineering Failure Analysis*, *169*, 109185.

[9] Keßler, J., Pelka, C., & Marx, S. (2025). Preservation of Masonry Arch Bridges in the Network of Deutsche Bahn. In *International Brick and Block Masonry Conference* (pp. 540-555). Springer, Cham.

[10] Bozyigit, B., & Acikgoz, S. (2022, November). Dynamic amplification in masonry arch railway bridges. In *Structures* (Vol. 45, pp. 1717-1728). Elsevier.

[11] Pantò, B., Grosman, S., Macorini, L., & Izzuddin, B. A. (2022). A macro-modelling continuum approach with embedded discontinuities for the assessment of masonry arch bridges under earthquake loading. *Engineering Structures*, *269*, 114722.

[12] Bagherzadeh Azar, A., & Sari, A. (2024). Failure analysis and structural resilience of a masonry arch Bridge subjected to blast loads: The Case study, Halilviran Bridge. *Mechanics of Advanced Materials and Structures*, 1-24.

[13]Vojislav Kecman: "Learning and Soft Computing — Support Vector Machines, Neural Networks, Fuzzy Logic Systems", The MIT Press, Cambridge, MA, 2001

[14] Tapkın, S., Tercan, E., Motsa, S. M., Drosopoulos, G., Stavroulaki, M., Maravelakis, E., & Stavroulakis, G. (2022). Structural investigation of masonry arch bridges using various nonlinear finite-element models. *Journal of Bridge Engineering*, *27*(7), 04022053.

[15] Silva, R., Costa, C., & Arêde, A. (2022, May). Numerical methodologies for the analysis of stone arch bridges with damage under railway loading. In *Structures* (Vol. 39, pp. 573-592). Elsevier.

Journal of Optimization of Soft Computing (JOSC)

Vol. 2, Issue 4, pp: (28-39), Winter-2024
Journal homepage: https://sanad.iau.ir/journal/josc

Paper Type (Research paper)

Edge-based Object Detection using Optimized Tiny YOLO on Embedded Systems

Peyman Babaei

Department of Computer Engineering, West Tehran Branch, Islamic Azad University, Tehran, Iran.

Article Info

Article History:
Received: 2024/12/29
Revised: 2025/01/30
Accepted: 2025/03/05

DOI:

Keywords:
Tiny YOLO, Model optimization, Model Deployment, Quantization, Pruning, Weight Clustering, Embedded Systems.

**Corresponding Author's Email Address: Peyman.Babaei@IAU.ac.ir*

Abstract

Object detection at the edge has gained considerable attention for enabling real-time, low-latency, and privacy-preserving solutions by processing data locally on resource-constrained devices. This paper explores using Tiny YOLO, a lightweight variant of the YOLO architecture, for object detection on embedded systems. Tiny YOLO is specifically designed for edge devices to run efficiently on constrained devices by utilizing a reduced architecture with fewer parameters while maintaining good performance for real-time object detection. The study examines the deployment of optimized Tiny YOLO models on embedded systems, incorporating techniques like quantization, pruning, and clustering to reduce model size, enhance speed, and lower power consumption. Optimization methods show significant improvements, with quantization speeding up inference, pruning eliminating redundant parameters, and clustering enhancing accuracy. Specifically, the study compares the performance of Tiny YOLO under these optimization techniques, presenting results for both Pascal VOC and COCO datasets. The results demonstrate that optimized Tiny YOLO models are effective for real-time object detection on microcontrollers. These methods enable the efficient deployment of deep learning models for edge computing, without relying on cloud infrastructure.

1. Introduction

In recent years, edge computing has emerged as a key technology for processing data closer to where it is generated, offering distinct advantages over traditional cloud-based computing. At its core, edge computing allows devices to process and analyze data locally rather than sending it to centralized servers or the cloud [1]. This localized processing significantly reduces latency, decreases reliance on network bandwidth, improves privacy, and increases overall system efficiency, making it particularly valuable for real-time applications such as image classification and object detection.

Embedded systems, which are small, low-cost, and energy-efficient computing units, are a key enabler of edge computing and Internet of Things devices. They are commonly used in applications where space and power consumption are constrained, such as in smart home devices, wearable electronics, and industrial sensors. However, microcontrollers are typically limited in terms of computational power, memory, and storage, making it challenging to run complex machine learning models [2,3].

Traditional deep learning models require substantial computational resources, especially in terms of processing power and memory, which makes it difficult to deploy them on embedded systems. However, recent advancements in model optimization techniques, such as quantization, pruning, and the use of lightweight neural network architectures (e.g., Tiny YOLO), have made it possible to deploy deep learning-based object detection models even on microcontrollers. These optimization techniques help reduce the size of the models, increase their inference speed, and reduce power consumption, all while maintaining acceptable levels of accuracy.

Deploying deep learning models on embedded systems is a key step in bringing artificial intelligence to the edge, where real-time decision-making is critical [4,5]. While challenges such as limited computational power, memory, and energy resources remain, advancements in model optimization techniques, lightweight architectures, and specialized hardware accelerators are making AI deployment on small devices more feasible [6,7]. For example, Tiny YOLO, a compact version of the well-known YOLO (You Only Look Once) object detection model, has proven to be effective for edge deployment due to its small size and efficient performance. This is especially valuable in applications such as autonomous systems, security surveillance, and robotics, where real-time object detection is needed on resource-constrained devices. One of the key hurdles in deploying deep learning on embedded systems is ensuring that these models can operate efficiently while maintaining a balance between performance and resource consumption [8-10].

Model optimization methods like quantization, pruning, and clustering help in reducing the memory footprint, lowering computation requirements, and speeding up inference times, making these models more suitable for edge devices like ESP32 [11]. Tools such as TensorFlow Lite provide frameworks that make it easier to run AI models on these constrained platforms, optimizing them further for mobile and embedded applications [12].

The rise of AI-powered microcontrollers is transforming industries by enabling smarter, decentralized systems [13,14]. In smart homes, microcontrollers are being used for voice recognition in virtual assistants and object detection in security cameras. In healthcare, wearable devices equipped with AI can monitor vital signs and detect falls in real-time. In industrial IoT, microcontrollers power predictive maintenance systems that can analyze sensor data like vibration and temperature to prevent equipment failure. Additionally, environmental monitoring using microcontrollers allows for the processing of data to predict weather patterns, track pollution levels, and monitor wildlife. The agricultural sector benefits from AI-enabled microcontrollers by enabling crop health monitoring, soil condition analysis, and pest detection, ultimately advancing precision farming techniques [15,16]. These examples underscore the versatility of microcontroller-based AI, showcasing its potential to enhance various domains by making intelligent decisions at the edge [17,18].

This study conducted aimed to evaluate the performance of the Tiny YOLO model on various edge devices, including ESP32, ESP32-S3, Pico W, and Jetson Nano, across different optimization techniques such as quantization, weight pruning, and clustering. The experiment utilized the COCO [19] and Pascal VOC [20] datasets to assess the model's mean Average Precision (mAP), frames per second (FPS), model size, inference time. Results showed that while ESP32 and Pico W exhibited significant limitations in accuracy and real-time performance due to their limited computational power, applying optimizations did provide some improvements in terms of model size and inference speed. In contrast, Jetson Nano demonstrated superior performance, achieving high mAP values and fast inference times, even with optimized models. This highlighted the importance of hardware capabilities in achieving real-time object detection, with Jetson Nano proving to be the most suitable platform for running optimized models like Tiny YOLO efficiently on more complex datasets.

In the following, the Edge-based object detection is presented in section 2, the YOLO and Tiny YOLO architectures are presented in sections 3 and 4. The optimization techniques of learning models are presented in section 5, which also refers to the proposed approach. In section 6, the implementation of different scenarios of Tiny YOLO model optimization are presented, and then in section 7, the results of evaluation are compared. Finally, the conclusion is presented in section 8.

2. Edge-based object detection

Deploying object detection models on embedded systems for edge computing is a promising solution for a wide range of real-time applications. As optimization techniques improve, the ability to run sophisticated object detection algorithms on embedded systems will continue to advance, opening up new possibilities in fields such as healthcare, security, autonomous systems, and environmental monitoring. The ability to perform local image processing without relying on cloud infrastructure is transforming industries and enabling more intelligent, responsive, and energy-efficient systems.

This breakthrough allows for real-time object detection on devices with limited resources. The ability to process images and classify objects at the edge, without the need for cloud computing, opens up a wide range of possibilities for various applications [21-23]. Below are some key use cases where microcontroller-based image processing is particularly beneficial:

- **Smart Home Automation:**
  - **Object Detection:** Embedded systems can be used to deploy object detection models to detect objects, faces, or gestures in smart home environments. For example, a security camera system could use a microcontroller to classify objects in real-time, identifying potential intruders or monitoring for specific actions.
  - **Gesture Recognition:** In a smart home, gesture recognition can be used to control lighting or appliances with simple hand movements, all processed on an embedded system.
- **Healthcare and Medical Devices:**
  - **Medical Imaging:** Embedded systems can assist in analyzing medical images such as X-rays, CT scans, or skin lesions directly on medical devices, facilitating faster diagnosis and reducing the need for data transmission to the cloud.
  - **Wearable Health Devices:** Image classification models deployed on wearable devices can monitor the health of individuals by identifying changes in skin tone, detecting the presence of medical conditions, or tracking movement patterns for rehabilitation purposes.
- **Industrial Automation and Monitoring:**
  - **Defect Detection in Manufacturing:** Embedded systems with object detection capabilities can be used in automated inspection systems to identify defects in products on an assembly line, improving quality control and reducing human error.
  - **Predictive Maintenance:** By analyzing visual data from sensors, embedded systems can help detect signs of wear or malfunction in machinery, enabling predictive maintenance and preventing downtime.
- **Autonomous Systems:**
  - **Robotics:** Autonomous robots, drones, and vehicles can leverage image classification at the edge to understand and interpret their environment, recognizing obstacles, people, or objects in real-time for navigation and decision-making.
  - **Agriculture and Environmental Monitoring:** Drones equipped with embedded systems can analyze images of crops or forests to monitor plant health, detect diseases, and evaluate environmental conditions without needing cloud-based processing.
- **Smart Cities and Surveillance:**
  - **Public Safety and Security:** Microcontrollers embedded in surveillance cameras can perform face recognition or detect unusual behaviors, enabling automated security systems that operate in real-time without relying on cloud servers.
  - **Traffic Monitoring:** Embedded systems can be used in traffic cameras to analyze road conditions, detect traffic congestion, or recognize vehicle types, all processed locally for faster decision-making.
- **Environmental Monitoring:**
  - **Wildlife Monitoring:** Edge devices equipped with embedded systems can monitor wildlife, detecting and identifying animals in remote areas through camera traps, without needing to transmit large image files to the cloud.
  - **Pollution Detection:** Image classification models can help detect pollution or other environmental hazards through cameras, enabling automated monitoring systems for air, water, or land quality.
- **Retail and Consumer Interaction:**
  - **Product Recognition:** Embedded systems can be used in point-of-sale systems or vending machines to recognize products through image classification, enabling automatic stock tracking or facilitating seamless customer interactions.
  - **Customer Behavior Analysis:** In retail settings, embedded systems can process visual data from in-store cameras to track customer behavior, optimize store layouts, or improve marketing strategies based on customer interaction patterns.

3. YOLO architectures

YOLO (You Only Look Once) is a popular series of deep learning models for object detection. It’s known for its speed and efficiency, making it a best choice for real-time object detection tasks. Over the years, different versions of YOLO have been released, each with improvements in accuracy, speed, and architecture [24,25]. The summary of YOLO’s evolution is shown in table 1. Below is an overview of the main versions and their key features:

- **YOLOv1**, introduced the idea of using a single convolutional neural network to predict bounding boxes and class probabilities in one pass, making it incredibly fast for real-time detection.
  - **Architecture:** A single convolutional neural network that simultaneously predicts bounding boxes and class probabilities for all objects in the image in one evaluation. The network

divides the image into a grid and for each grid cell, it predicts:
- Bounding boxes (x, y, width, height)
- Confidence score (how likely the box contains an object)
- Class probabilities (which object class the box belongs to).

- **Strengths:** Very fast (real-time detection), unified approach (object localization and classification in one pass).
- **Weaknesses:** Struggles with detecting small objects and handling overlapping objects, less accurate in comparison to other models like Faster R-CNN.

❖ **YOLOv2**, released in 2017, brought significant improvements such as the introduction of anchor boxes, batch normalization, and multi-scale training, which increased both speed and accuracy, especially for larger objects.

- **Architecture:**
  - Introduced improvements like a new backbone network, Darknet-19, which was more powerful than YOLOv1's architecture.
  - Added anchor boxes for better bounding box prediction, addressing the issue of poor localization seen in YOLOv1.
  - Used multi-scale training, where the model was trained on different image sizes to improve generalization.
  - Introduced batch normalization to stabilize and speed up training.
- **Strengths:** Faster and more accurate than YOLOv1, improved handling of different object scales, better generalization, and more robust performance.
- **Weaknesses:** Still struggles with small object detection.

❖ **YOLOv3**, released in 2018, the model was further enhanced with a new backbone (Darknet-53), multi-label classification, and the use of three different scales for prediction, allowing it to better detect small objects. Despite these improvements, YOLOv3 still had limitations when compared to more complex models like Faster R-CNN.

- **Architecture:**
  - YOLOv3 used a new backbone called Darknet-53, which improved accuracy and allowed for better feature extraction.
  - Used multi-label classification to improve the detection of objects with more than one class.
  - Introduced three different scales for prediction (small, medium, and large), allowing the network to detect objects at various sizes.
  - Introduced Residual Connections to help deeper networks train better and avoid vanishing gradients.
  - The output layer was redesigned to use logistic regression for bounding box prediction.
- **Strengths:** Better detection of smaller objects, significant performance improvement over v2 in terms of both speed and accuracy.
- **Weaknesses:** Still not as accurate as more complex architectures like Faster R-CNN for certain tasks, especially in cases of very dense or small objects.

❖ **YOLOv4** released in 2020, focused on improving detection performance with a new backbone (CSPDarknet53) and techniques like Mosaic data augmentation and self-adversarial training, leading to better accuracy, especially for small and dense objects, while maintaining fast inference times.

- **Architecture:**
  - Built on the YOLOv3 model but incorporated several new techniques for better performance, including:
    - CSPDarknet53 as the backbone network, which balances accuracy and speed.
    - Mosaic Data Augmentation to improve generalization by combining multiple images during training.
    - Self-adversarial training for improved robustness.
    - DropBlock regularization for better bounding box predictions.
  - Improved performance on smaller objects with better feature pyramids.
- **Strengths:** Higher accuracy than YOLOv3, better at handling small and dense objects, faster inference times, state-of-the-art performance in real-time detection.
- **Weaknesses:** Larger model size compared to earlier versions, requiring more computational resources.

❖ **YOLOv5**, which was not developed by the original YOLO creators but became very popular due to its ease of use, modular design, and efficient performance on a range of hardware.

- **Architecture:**
  - YOLOv5 is a separate project developed by Ultralytics, which is not an official continuation of the YOLO series but has become very popular in the community.

- It focuses on speed and ease of use, and its codebase is built in PyTorch (as opposed to Darknet for the official YOLO models).
- YOLOv5 uses a modular design with different model sizes (small, medium, large, extra-large) to balance speed and accuracy.

o **Strengths:** Very easy to use, with a lot of built-in features like model training, testing, and deployment. Achieves state-of-the-art performance with relatively lightweight models.

o **Weaknesses:** It is not an official release from the original YOLO authors, so it may differ in implementation or long-term support compared to the official YOLO versions.

❖ **YOLOv6**, released in 2022, continued the trend of optimization, especially for edge devices, by focusing on speed and efficiency.

o **Architecture:**
- YOLOv6 is optimized for both speed and accuracy with improvements over YOLOv5, particularly in handling dense and small objects.
- Introduced a more efficient backbone (CSPResNet) and neck (PP-YOLO) to enhance detection performance.
- Focused on optimizing inference speed for deployment on edge devices.

o **Strengths:** Real-time performance, better accuracy with fewer resources.

o **Weaknesses:** Like YOLOv5, it's not an official version, so community-driven development may lead to less consistency over time.

❖ **YOLOv7**, also released in 2022, utilized more advanced techniques such as efficient transformers and heterogeneous module fusion, further enhancing both speed and accuracy.

o **Architecture:**
- YOLOv7 continues improving on YOLOv5 and YOLOv6, focusing on both accuracy and inference speed. It utilizes the efficient transformer architecture for better handling of spatial relationships in images.
- Improved backbone for better feature extraction and information flow.
- Introduced Heterogeneous Module fusion for better performance in terms of both accuracy and speed.

o **Strengths:** One of the fastest YOLO versions to date, highly optimized for real-time object detection.

o **Weaknesses:** Complexity in tuning for specific tasks, requires careful hyperparameter tuning for optimal performance.

❖ **YOLOv8**, introduced in 2023, offers cutting-edge performance with improvements in backbone architectures, better handling of various object detection tasks, and optimization for real-time and embedded systems.

o **Architecture:**
- YOLOv8 aims to offer even better accuracy, speed, and efficiency than its predecessors. It is designed to perform well on various object detection tasks and includes newer backbone and neck architectures, as well as better loss functions for bounding box predictions.
- It also focuses on fine-tuning for specific tasks like segmentation and key point detection.

o **Strengths:** Cutting-edge performance, high accuracy, and optimized for both real-time and edge devices.

o **Weaknesses:** Requires more computational resources than earlier versions but offers a significant boost in performance.

Table 1: Summary of YOLO's evolution.

Version	Key Features
YOLOv1	First release; groundbreaking for real-time object detection using a single CNN for bounding box and classification predictions.
YOLOv2	Improved accuracy and speed; introduced anchor boxes, batch normalization, and multi-scale training. Better at handling larger objects.
YOLOv3	Significant improvements in architecture with Darknet-53 backbone; better at detecting small objects with multi-scale predictions and multi-label classification.
YOLOv4	Focused on speed, accuracy, and robustness, especially for real-time applications; introduced Mosaic data augmentation and CSPDarknet53.
YOLOv5	A community-driven model; emphasizes ease of use, modular design, and optimized for both speed and accuracy, with multiple model sizes.
YOLO v6 & v7	Optimized for edge devices and real-time applications; further enhancements in speed, accuracy, and performance, especially in dense or small object detection.
YOLOv8	The latest version with cutting-edge performance and optimizations for real-time and embedded devices; handles various detection tasks.

The YOLO family continues to evolve with a stronger emphasis on speed, accuracy, and resource efficiency, making it a top choice for real-time object detection in areas like autonomous driving, surveillance and robotics. Each version of YOLO has brought improvements in terms of accuracy, speed, and efficiency, making it one of the top choices for real-time object detection in fields such as autonomous driving, robotics, and surveillance.

4. Tiny YOLO

Tiny YOLO is a smaller, lighter version of the YOLO model, specifically designed for applications where computational resources are limited, such as on edge devices or in real-time systems that require fast processing speeds. It is a trade-off between performance and efficiency, sacrificing some accuracy for the sake of reduced size and faster inference time. Tiny YOLO simplifies the architecture of the original YOLO by reducing the number of layers and parameters. For example, in Tiny YOLO, the backbone network (typically Darknet) has fewer convolutional layers and a smaller number of filters. This results in faster processing speeds and reduced memory requirements, making it suitable for devices with limited computational power, such as embedded systems, mobile devices, and IoT applications.

Faster Inference: Tiny YOLO is much faster than the standard YOLO models due to its smaller size and fewer parameters. This makes it ideal for real-time object detection applications, especially on resource-constrained devices.

Lower Computational Requirements: The reduced architecture allows Tiny YOLO to run efficiently on devices with limited GPU or CPU capabilities. It's particularly useful for edge devices, mobile phones, and embedded systems where processing power is a concern.

Smaller Model Size: The smaller model size makes it easier to deploy Tiny YOLO on devices with limited storage capacity. This is important for applications where storage space is constrained, such as drones or IoT devices.

Good for Low-Latency Applications: Because of its faster processing, Tiny YOLO is suited for low-latency tasks where quick decision-making is necessary, such as autonomous vehicles or real-time video surveillance.

Lower Accuracy: Because of the simplified architecture, Tiny YOLO generally achieves lower accuracy compared to full YOLO versions (like YOLOv3, YOLOv4, or YOLOv5). It may struggle with detecting small objects or complex scenes with a high degree of clutter.

Limited Detection Capabilities: While Tiny YOLO is good for general object detection, its performance can degrade in challenging scenarios, such as detecting objects in high-density environments or cases where fine-grained classification is required.

Less Robust in Difficult Conditions: Tiny YOLO might not perform as well under varying conditions, such as different lighting, weather, or occlusion, compared to more complex models.

Tiny YOLO is a powerful tool when you need object detection on devices with limited resources, where speed and efficiency are more critical than achieving the highest possible accuracy. Its trade-off between performance and resource usage makes it suitable for real-time applications like autonomous vehicles, drones, and mobile devices.

Key Characteristics of Tiny YOLO's Architecture are:

- **Fewer layers and filters:** The network has fewer layers and smaller filter sizes compared to the full YOLO versions, making it faster but less accurate.
- **Simplified structure:** By reducing the depth of the network and the number of neurons in the fully connected layers, Tiny YOLO is optimized for speed and smaller model size.
- **Max Pooling:** Max pooling layers help reduce the spatial resolution of feature maps, aiding in faster processing and reducing overfitting by discarding irrelevant details.
- **Lower resolution input:** Tiny YOLO generally works with lower resolution input images, which reduces computation time but may decrease accuracy in detecting small objects.

Tiny YOLO sacrifices some complexity and accuracy from the standard YOLO architecture in exchange for faster processing and reduced computational requirements. This makes it suitable for real-time applications on edge devices and embedded systems, where speed and low resource consumption are prioritized over the highest possible accuracy. The Tiny YOLO architecture table is shown in table 2. The layers of this architecture are described below:

Input Layer: Takes images of size 224x224x3, commonly used for image classification and detection tasks.

Convolutional Layers: These layers progressively extract more abstract features from the image by applying convolution with 3x3 filters. The number of filters increases as the network deepens, allowing for more complex representations.

Max Pooling Layers: Reduce the spatial dimensions of the feature maps, making the model more efficient and helping to avoid overfitting.

Fully Connected Layers: Compress the features extracted from the convolutional layers and map them to a higher-dimensional space, enabling the prediction of object classes and bounding boxes.

Output Layer: Predicts both the class probabilities and bounding box positions (class + 4 for bounding box coordinates). The final output is structured to handle N classes and the corresponding bounding box for each object detected.

Table 2: Tiny YOLO architecture.

Layer	Number of Filters	Filter Dimensions	Output Dimensions
Input Layer	---	224x224	224x224x3
Convolutional 1	16	3x3	224x224x16
MaxPooling 1	---	2x2	112x112x16
Convolutional 2	32	3x3	112x112x32
MaxPooling 2	---	2x2	56x56x32
Convolutional 3	64	3x3	56x56x64
MaxPooling 3	---	2x2	28x28x64
Convolutional 4	128	3x3	28x28x128
MaxPooling 4	---	2x2	14x14x128
Convolutional 5	256	3x3	14x14x256
MaxPooling 5	---	2x2	7x7x256
Fully Connected 1	4096	N/A	1x1x4096
Fully Connected 2	Classes + 4	N/A	1x1x(N+4)
Output	N/A	N/A	1x1x(N+4)

This structure is a simplified version of the YOLO architecture, designed for efficient image classification and object detection with reduced computational resources.

5. Model Optimization Techniques

Model optimization techniques aim to reduce the size and computational demands of machine learning models without compromising their performance. This is crucial for deploying models on small, resource-limited devices. Methods such as pruning, quantization, and weight clustering are commonly used to achieve this goal [26]. The main objective is to enable large models to run smoothly on edge devices with limited memory, processing power, and battery life. These optimizations are especially useful for applications requiring continuous operation. The benefits of using optimization techniques include:

Inference Speed: Large models take longer to make predictions, which can be problematic for real-time applications like video or audio processing. Optimization enhances inference speed, making models more suitable for time-sensitive tasks.

Cost and Resource Efficiency: Training and deploying large models demand substantial computational resources, often resulting in high costs. Optimization reduces these needs, enabling faster and more efficient training and deployment.

Deployment Flexibility: Large model sizes can hinder deployment on certain platforms or environments. Optimization makes models more portable and easier to deploy.

Quantization is a technique that reduces the size and computational complexity of machine learning models by using fewer bits to represent weights and activations. It is particularly useful for devices with limited memory and computational power, like edge and IoT devices. The technique involves reducing the precision of model weights, such as converting 32-bit floating-point numbers to 8-bit integers, which reduces model size and improves inference speed but may slightly affect accuracy. Quantization can be applied during or after training, with post-training quantization being simpler but potentially introducing errors, while quantization-aware training simulates quantization effects during training to preserve accuracy and improve performance. The main benefits include faster inference, reduced memory use, and lower energy consumption, but balancing model size and accuracy requires careful calibration [27,28].

Pruning is a method used to reduce model size by removing unnecessary parameters, lowering computational and storage needs, and improving generalization. It involves setting certain weights to zero, thus removing them from the model. Pruning can be done before, during, or after training and is effective for various models like deep neural networks and decision trees. The benefits of pruning include reduced size, simpler interpretation, and easier deployment. Weight pruning is commonly used, where less important weights are set to zero, creating sparsity in the model and reducing memory usage. While it speeds up inference, excessive pruning may degrade performance, requiring a balance between model size and accuracy [29,30].

Weight clustering is another optimization technique that reduces the number of unique weight values in a model. Instead of storing each individual weight, only unique values are saved, minimizing memory usage. The technique groups similar weights into clusters, often using the cluster centroid as the representative value for all weights in that group. By reducing the number of clusters, the model becomes more compact, saving memory and improving efficiency [31].

6. Implementation of Optimized Models

The objective of this experiment was to evaluate the deployment performance of the Tiny YOLO model on various embedded hardware platforms, including the ESP32, ESP32-S3, Pico W, and Jetson Nano. These platforms were chosen to compare the feasibility of running a real-time object detection model like Tiny YOLO on resource-constrained devices, with a focus on the impact of optimization techniques such as quantization, weight pruning, and clustering.

The ESP32 and Pico W are microcontroller-based platforms known for their low power consumption and small form factors, making them suitable for simple edge applications. However, their limited computational power and memory impose constraints when running more complex deep learning models like Tiny YOLO. The ESP32-S3 variant was also included in the test, which offers enhanced AI capabilities compared to the basic ESP32 model, but still lacks the computational resources required for high-performance tasks. These microcontrollers were tested with optimizations to reduce the size of the model, improve inference time, and reduce latency. Quantization was used to reduce the precision of weights and activations, weight pruning removed less important parameters to decrease model size, and clustering grouped similar weights to further optimize the model.

The Jetson Nano, a more powerful platform equipped with a GPU and designed specifically for AI applications, was also tested. It provides significant computational power, making it better suited for real-time deep learning tasks. The Jetson Nano was used as a benchmark to compare the performance of the microcontroller-based platforms and to see how well Tiny YOLO can perform with more robust hardware. The same optimization methods were applied to the Jetson Nano to assess their impact on performance, although the higher computational power of the device meant that the benefits of optimization were less significant than on the microcontrollers.

The following metrics were measured across all devices: mean Average Precision, Frames Per Second, Model Size, Inference Time, and Latency. These metrics were used to evaluate the trade-offs between performance and computational efficiency after applying the optimization techniques. In the case of ESP32, ESP32-S3, and Pico W, the models were optimized to fit within the limited memory constraints of the devices. The resulting models were small in size but showed significant limitations in terms of accuracy, speed, and real-time performance, as the inference time remained high.

Overall, this experiment demonstrated that while optimizations such as quantization, pruning, and clustering can help make deep learning models more feasible for microcontroller-based platforms, the limited computational power of devices like ESP32 and Pico W remains a major bottleneck for real-time object detection tasks. On the other hand, the Jetson Nano proved to be a much more capable platform for deploying Tiny YOLO in real-time applications.

Quantization is first applied by converting the model’s 32-bit floating-point weights and activations to 8-bit integers. This reduces the model's size and boosts inference speed. The model is then assessed for memory savings, computational efficiency, and any slight loss in accuracy due to the reduction in numerical precision. Next, pruning is performed by eliminating weights that have little impact on the model’s performance during training, thus reducing both the model size and computational load. The pruned model is tested to evaluate the balance between efficiency improvements and any potential accuracy loss, which depends on the extent of pruning. Lastly, weight clustering is implemented, grouping similar weights into a predefined number of clusters and replacing them with shared centroids. This technique reduces memory usage without affecting numerical precision, and the clustered model is assessed for memory savings and any accuracy degradation caused by reduced weight granularity.

Deploying optimized models on hardware platforms like ESP32, ESP32-S3, Pico W, and Jetson Nano offers a range of possibilities, each suited to different use cases based on the computational power and application requirements. By applying techniques like quantization and pruning, the model's size and inference time can be reduced, making it more feasible for deployment on edge devices. Overall, selecting the appropriate platform depends on the balance between performance, power consumption, and the complexity of the task at hand.

7. Evaluation Results

Performance of each optimized model is compared to the base model to evaluate the benefits and trade-offs of each technique. The results of the combined optimization methods are also analyzed to find the best strategy for balancing performance and efficiency. This evaluation provides valuable insights for deploying Tiny YOLO in real-world scenarios with limited resources. The evaluation focuses on key metrics such as mean Average Precision (mAP), Frames Per Second (FPS), and Inference Time (ms), which collectively assess the models' performance and suitability for resource-constrained environments. When deploying Tiny YOLO on embedded systems, it's essential to consider various metrics. These metrics help understand the trade-offs between efficiency and accuracy, guiding the optimization process.

Table 3 focuses only on the Pascal VOC dataset for the Tiny YOLO models deployed on ESP32, ESP32-S3, Pico W, and Jetson Nano, providing a comprehensive framework for evaluating the optimized Tiny YOLO models. The models balance high accuracy with smaller size, improved efficiency, and reduced inference time, making them suitable for image classification tasks in resource-limited environments.

Table 3: Evaluation results for Pascal VOC dataset.

optimization Method	Device	mAP (%)	FPS	Inference Time (ms)
Base Model	ESP32	35.2	1.5	2750
	ESP32-S3	27.3	2.5	1879
	Pico W	35.0	1.0	2940
	Jetson Nano	77.0	17.0	279
Quantization	ESP32	34.9	1.5	947
	ESP32-S3	27.1	2.5	738
	Pico W	33.8	1.0	1095
	Jetson Nano	76.7	17.0	127
Pruning	ESP32	34.5	1.5	1030
	ESP32-S3	26.8	2.5	712
	Pico W	33.6	1.0	1240
	Jetson Nano	76.5	17.0	145
Clustering	ESP32	34.2	1.5	968
	ESP32-S3	26.6	2.5	780
	Pico W	33.2	1.0	1155
	Jetson Nano	76.2	17.0	132

In terms of mean Average Precision (figure 1), ESP32 and Pico W show relatively low values, ranging from 34.2% to 35.2%, even after applying optimization techniques like quantization, pruning, and clustering. These platforms struggle to achieve high accuracy due to their limited processing power. On the other hand, Jetson Nano demonstrates significantly higher mAP values, ranging from 76.2% to 77%, which is a clear reflection of its superior computational capabilities. Despite optimizations, the Jetson Nano consistently maintains strong accuracy, making it a better choice for tasks requiring higher precision.

For inference time (figure 2), ESP32, ESP32-S3, and Pico W have high values, ranging from 712ms to 2940ms, due to their hardware constraints. This long inference time is detrimental to real-time object detection, as it introduces delays in processing. Conversely, Jetson Nano achieves much faster inference times, ranging from 127ms to 145ms, depending on the optimization method applied. This makes Jetson Nano an ideal platform for real-time object detection.

Jetson Nano outperforms ESP32 and Pico W across all evaluation metrics, including mAP, FPS, inference time, and latency, making it the best choice for real-time object detection tasks using Tiny YOLO. While ESP32 and Pico W offer low-cost and power-efficient solutions, their performance for complex models like Tiny YOLO is limited, making them unsuitable for real-time applications that require high accuracy and speed. Despite the modest improvements offered by optimization techniques such as quantization, pruning, and clustering, the hardware constraints of the microcontroller-based platforms continue to limit their ability to perform effectively for more demanding tasks.

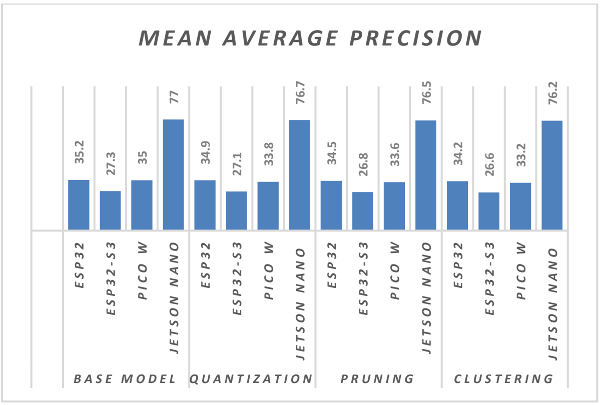


Figure 1: The mAP for Pascal VOC.

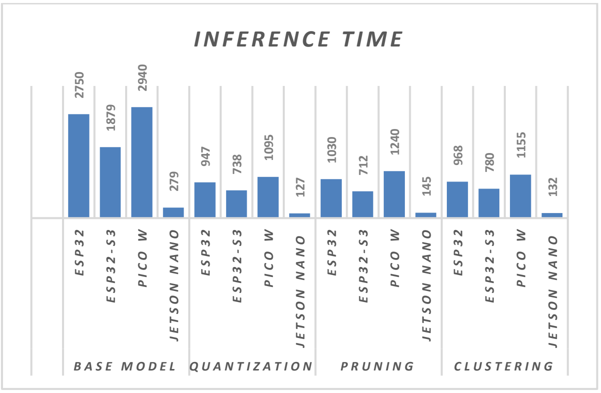


Figure 2: Inference time for Pascal VOC.

Table 4 focuses only on the COCO dataset for the Tiny YOLO models deployed on ESP32, ESP32-S3, Pico W, and Jetson Nano, providing a comprehensive framework for evaluating the optimized Tiny YOLO models. The models balance high accuracy with smaller size, improved efficiency, and reduced inference time, making them suitable for image classification tasks in resource-limited environments.

In terms of mean Average Precision (figure 3), ESP32 and Pico W show relatively low values, with the highest mAP reaching around 27.5% even after applying optimization techniques. The limited computational resources on these microcontrollers result in lower accuracy, which is a significant challenge despite the optimizations. In contrast, Jetson Nano consistently achieves much higher mAP values, ranging from 66.9% to 67.7%, demonstrating the platform's ability to handle more complex models like Tiny YOLO with greater precision due to its superior hardware capabilities.

Table 4: Evaluation results for COCO dataset.

Optimization Method	Device	mAP (%)	FPS	Inference Time (ms)
Base Model	ESP32	29.1	1.5	3142
	ESP32-S3	35.5	2.5	2057
	Pico W	27.2	1.0	3260
	Jetson Nano	67.7	17.0	325
Quantization	ESP32	28.6	1.5	1180
	ESP32-S3	34.6	2.5	875
	Pico W	26.4	1.0	1308
	Jetson Nano	67.5	17.0	117
Pruning	ESP32	28.2	1.5	1270
	ESP32-S3	33.9	2.5	913
	Pico W	25.8	1.0	1382
	Jetson Nano	66.9	17.0	166
Clustering	ESP32	28.9	1.5	1195
	ESP32-S3	35.1	2.5	897
	Pico W	26.8	1.0	1336
	Jetson Nano	67.6	17.0	132

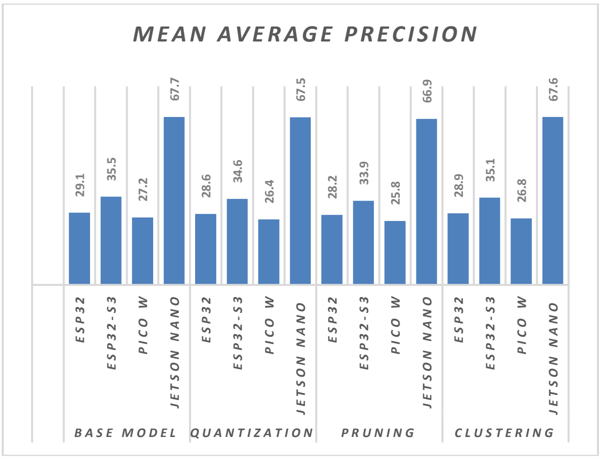


Figure 3: The mAP for COCO.

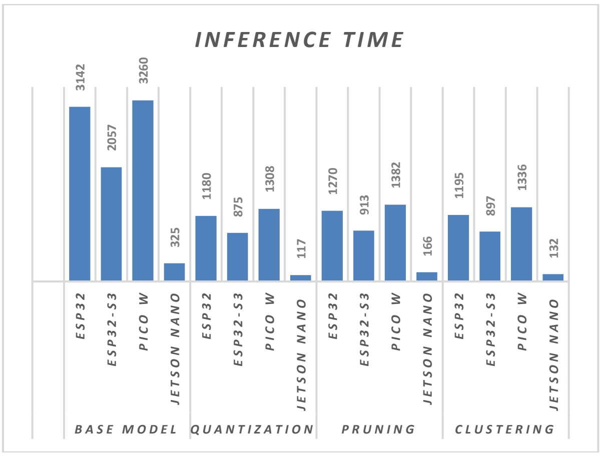


Figure 4: Inference time for COCO.

In terms of inference time (figure 4), ESP32, ESP32-S3, and Pico W exhibit high inference times ranging from 875ms to 3260ms, which makes these platforms unsuitable for real-time applications where speed is crucial. In contrast, Jetson Nano achieves much lower inference times, between 117ms and 166ms, making it well-suited for real-time tasks that demand faster processing.

Jetson Nano clearly outperforms both ESP32 and Pico W across all evaluation metrics, making it the optimal choice for real-time object detection with Tiny YOLO on the COCO dataset. The ESP32 and Pico W show significant limitations due to their hardware constraints, even after optimization, and are better suited for tasks of lower complexity or for applications where real-time performance is not as critical. These platforms can still be useful for simpler AI tasks, but when it comes to real-time detection requiring high accuracy, Jetson Nano is the clear leader.

8. Conclusion

The experiment conducted to evaluate the deployment of Tiny YOLO on a range of embedded systems, including ESP32, ESP32-S3, Pico W, and Jetson Nano, reveals key insights into the feasibility of running optimized deep learning models on resource-constrained devices. The evaluation was carried out on two popular object detection datasets, COCO and Pascal VOC, with the focus on the performance impact of three model optimization techniques: quantization, weight pruning, and clustering. The results, detailed in the tables, provide a comprehensive analysis of the trade-offs between mean Average Precision, frames per second, and inference time across different hardware platforms.

Jetson Nano, with its powerful GPU and higher computational resources, consistently outperformed the other platforms in terms of both mAP and real-time performance. This was expected, as the Jetson Nano is designed for AI applications, offering substantial processing power and memory to handle complex models like Tiny YOLO. It demonstrated an impressive mAP of around 66.9% to 67.7% on the COCO dataset, which is a significant advantage for more computationally intensive tasks. The inference time was also much lower compared to the microcontroller-based platforms, further emphasizing its suitability for real-time applications. However, optimizations like quantization, pruning, and clustering did lead to slight improvements in inference time and latency, showing that resource-efficient techniques can make these platforms viable for simpler tasks.

One notable aspect of the experiment is the importance of model optimization. While the optimizations did not dramatically increase the mAP on these low-power platforms, they did make the models more feasible for deployment, balancing the trade-off between computational efficiency and accuracy.

The results underscore the importance of selecting the right hardware for edge AI deployment, where a balance between computational power, model size, inference time, and energy consumption must be considered. Future work could focus on further optimizing the Tiny YOLO model for even smaller and more power-efficient devices while maintaining reasonable accuracy for a broader range of real-world applications.

References

[1] Kotha, H.D. and Gupta, V.M., 2018. IoT application: a survey. Int. J. Eng. Technol, 7(2.7), pp.891-896.
[2] Dian, F.J., Vahidnia, R. and Rahmati, A., 2020. Wearables and the Internet of Things (IoT), applications, opportunities, and challenges: A Survey. IEEE access, 8, pp.69200-69211.
[3] Asghari, P., Rahmani, A.M. and Javadi, H.H.S., 2019. Internet of Things applications: A systematic review. Computer Networks, 148, pp.241-261.
[4] Li, H., Ota, K. and Dong, M., 2018. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. IEEE network, 32(1), pp.96-101.
[5] Liangzhen Lai and Naveen Suda. 2018. Enabling Deep Learning at the IoT Edge. In Proceedings of the International Conference on Computer-Aided Design (San Diego, California) (ICCAD ’18). ACM, New York, NY, USA, Article 135, 6 pages.
[6] Singh, R. and Gill, S.S., 2023. Edge AI: a survey. Internet of Things and Cyber-Physical Systems, 3, pp.71-92.
[7] Wang, X., Han, Y., Leung, V.C., Niyato, D., Yan, X. and Chen, X., 2020. Edge AI: Convergence of edge computing and artificial intelligence (pp. 3-149). Singapore: Springer.
[8] David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T. and Warden, P., 2021. Tensorflow lite micro: Embedded machine learning for tinyml systems. Proceedings of Machine Learning and Systems, 3, pp.800-811.
[9] Rashidi, M., 2022. Application of TensorFlow lite on embedded devices: A hands-on practice of TensorFlow model conversion to TensorFlow Lite model and its deployment on Smartphone to compare model’s performance.
[10] Mamtha, G.N., Sharma, S. and Sing, N., 2023, December. Embedded Machine Learning with Tensorflow Lite Micro. In 2023 International Conference on Power Energy, Environment & Intelligent Control (PEEIC) (pp. 1480-1483).
[11] Berthelier, A., Chateau, T., Duffner, S., Garcia, C. and Blanc, C., 2021. Deep model compression and architecture optimization for embedded systems: A survey. Journal of Signal Processing Systems, 93(8), pp.863-878.
[12] TensorFlow Lite, TensorFlow, 2021. Available online: https://www.tensorflow.org/lite
[13] Hua, H., Li, Y., Dong, N., Li, W. and Cao, J., 2023. Edge computing with artificial intelligence: A machine learning perspective. ACM Computing Surveys, 55(9), pp.1-35.
[14] Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S. and Zomaya, A.Y., 2020. Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet of Things Journal, 7(8), pp.7457-7469.
[15] Grzesik, P. and Mrozek, D., 2024. Combining Machine Learning and Edge Computing: Opportunities, Challenges, Platforms, Frameworks, and Use Cases. Electronics, 13(3), p.640.
[16] Li, H., Ota, K. and Dong, M., 2018. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. IEEE network, 32(1), pp.96-101.
[17] Chang, Z., Liu, S., Xiong, X., Cai, Z. and Tu, G., 2021. A survey of recent advances in edge-computing-powered artificial intelligence of things. IEEE Internet of Things Journal, 8(18), pp.13849-13875.
[18] Sivaganesan, D., 2019. Design and development ai-enabled edge computing for intelligent-iot applications. Journal of trends in Computer Science and Smart technology (TCSST), 1(02), pp.84-94.
[19] Jain, S., Dash, S. and Deorari, R., 2022, October. Object detection using coco dataset. In 2022 International Conference on Cyber Resilience (ICCR) (pp. 1-4). IEEE.
[20] Shetty, S., 2016. Application of convolutional neural network for image classification on Pascal VOC challenge 2012 dataset. arXiv preprint arXiv:1607.03785.
[21] Li, C., Wang, J., Wang, S. and Zhang, Y., 2024. A review of IoT applications in healthcare. Neurocomputing, 565, p.127017.
[22] Afzal, B., Umair, M., Shah, G.A. and Ahmed, E., 2019. Enabling IoT platforms for social IoT applications: Vision, feature mapping, and challenges. Future Generation Computer Systems, 92, pp.718-731.
[23] Dian, F.J., Vahidnia, R. and Rahmati, A., 2020. Wearables and the Internet of Things (IoT), applications, opportunities, and challenges: A Survey. IEEE access, 8, pp.69200-69211.
[24] Tripathi, A., Gupta, M.K., Srivastava, C., Dixit, P. and Pandey, S.K., 2022, December. Object detection using YOLO: A survey. In 2022 5th International Conference on Contemporary Computing and Informatics (IC3I) (pp. 747-752). IEEE.
[25] Hussain, M., 2024. Yolov1 to v8: Unveiling each variant–a comprehensive review of yolo. IEEE Access, 12, pp.42816-42833.
[26] Babaei, P., 2024, March. Convergence of Deep Learning and Edge Computing using Model Optimization. In 2024 13th Iranian/3rd International Machine Vision and Image Processing Conference (MVIP) (pp. 1-6). IEEE.
[27] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W. and Keutzer, K., 2022. A survey of quantization methods for

efficient neural network inference. In Low-Power Computer Vision (pp. 291-326). Chapman and Hall/CRC.
[28] Rokh, B., Azarpeyvand, A. and Khanteymoori, A., 2023. A comprehensive survey on model quantization for deep neural networks in image classification. ACM Transactions on Intelligent Systems and Technology, 14(6), pp.1-50.
[29] Liang, T., Glossner, J., Wang, L., Shi, S. and Zhang, X., 2021. Pruning and quantization for deep neural network acceleration: A survey. Neurocomputing, 461, pp.370-403.
[30] Madnur, P.V., Dabade, S.H., Khanapure, A., Rodrigues, S., Hegde, S. and Kulkarni, U., 2023, November. Enhancing Deep Neural Networks through Pruning followed by Quantization Pipeline: A Comprehensive Review. In 2023 2nd International Conference on Futuristic Technologies (INCOFT) (pp. 1-8). IEEE.
[31] Choudhary, T., Mishra, V., Goswami, A. and Sarangapani, J., 2020. A comprehensive survey on model compression and acceleration. Artificial Intelligence Review, 53, pp.5113-5155.

Journal of Optimization of Soft Computing (JOSC)

Vol. 2, Issue 4, pp: (40-55), Winter-2024
Journal homepage: https://sanad.iau.ir/journal/josc

Paper Type (Research paper)

Investigating and sensitivity analyzing the of operating microgrids in the presence of electric vehicles

Yaser Ebazadeh[1], Reza Alayi[2*] and Eskandar Jamali[2]

1. *Department of Computer Engineering, Germi Branch, Islamic Azad University, Germi, Iran.*
2. *Department of Mechanics, Germi Branch, Islamic Azad University, Germi, Iran.*

Article Info

Article History:
Received: 2025/01/03
Revised: 2025/02/20
Accepted: 2025/03/05

DOI:

Keywords:
Renewable Energy, Optimization, Electrical Energy, Electric Vehicle.

**Corresponding Author's Email Address: reza.alayi@yahoo.com*

Abstract

The use of electric vehicles (EVs) is increasing daily owing to the lack of fossil fuels on the one hand and the emission of pollution caused by the burning of fossil fuels. In this research, to produce fuel for EVs from renewable sources, a power grid with renewable energy sources such as wind turbines and photovoltaic cells was considered. After modeling, optimization was performed to supply the energy needed by EVs using wind turbines and photovoltaic cells, and the main purpose of this optimization was to reduce environmental pollution by providing maximum energy. One of the important results is the $1601941 investment cost of the wind turbine and photovoltaic cell system.

1. Introduction

For a long time, the national electricity grid has been used to supply the electric energy needed by the cities, and the arrangement of the power plants has always been of special importance about the required loads. Become One of the solutions to reduce losses is the use of scattered production with the approach of using renewable energy sources, and with the production of electric vehicles, this device is also considered as a power grid. These effects can include an increase in the maximum load, an increase in losses, a decrease in the voltage and load factor of the system, etc. [1].

Various research has been done in the field of how to charge and discharge electrical vehicles by the power grid [2]. In reference [3], has provided voltage and energy control in distribution systems in the presence of flexible loads considering coordinated charging of EVs. The purpose of this method is to charge in low load hours with low energy prices and at the same time meet the technical limitations of the network. Search methods and neural networks are used to make decisions in this system. Reference [4] study provides a review of the existing coupler, compensation topologies, and control schemes to determine their effectiveness in achieving the desired control objectives. In addition, it introduces practical metrics for a system designer to consider when developing the magnetics and power electronics for a DWPT system to ensure good controllability. It also shows how the delay in communication can affect the control performance and impact recommendations for high-speed vehicle charging. In [5], this study analyses the effectiveness of an off-grid solar photovoltaic system for the charging of EVs in a long-term parking lot. The effectiveness of charging is investigated through analysis of the states of charge (SoC) at the departure of EVs plugged in at the parking lot over the simulated year.

References [6] and [7] have discussed the technical and economic impact of the introduction of EVs on the electric grid of the United States of America. These articles show that the increase in the arrival of vehicles can cause difficulty in the work of the network operator and lower the reliability of the network. In these articles, the use of intelligent charging planning and also vehicle-to-grid (V2G) capability is proposed as a way to overcome this problem. In [8] and [9], have provided inductive power transfer charging infrastructure for EVs: A New Zealand case study. Reference [10] has addressed the issue of the effect of EVs on the reliability of the distribution network. In this study, accumulators are considered battery exchange stations. Here, an algorithm is used that divides the time into different intervals in such a way that load fluctuations can be ignored. Accordingly, in each interval, the probability distribution function of the batteries' energy has been taken into account, taking into account the drivers' battery replacement pattern. In this article, the behavior of drivers is considered based on their behavior at gas stations. The study is done on a 34-bus system IEEE [10].

Reference [11] has provided a Technical Review of Advanced Approaches for EV Charging Demand Management, Part I: Applications in Electric Power Market and Renewable Energy Integration. In reference [12], presented the Impact of EV charging demand on distribution transformers in an office area and determination of flexibility potential. In Reference [13, 14], the coordinator tries to find the most optimal charging program for EVs by implementing an optimization problem with the objective function of minimizing network operation costs by satisfying the condition of supplying the load required by vehicles. In this article, both modes of modeling the coordinator as a price receiver and affecting the price offered to vehicle owners are considered.

An optimization-based approach is introduced in [15] to properly allocate multiple wind turbine generation systems (WTGS) in distribution systems in the presence of (plug-in electric vehicles) PEVs. The proposed approach considers 1) uncertainty models of WTGS, PEV, and loads, 2) DSTATCOM functionality of WTGS, and 3) various system constraints. In [16], a dual-solver framework based on model predictive control is proposed, E-solver and L-solver. The economic scheduling problem is formulated using mixed-integer linear programming, which can be solved efficiently way by using a commercial solver.

Recent research has shown that smart charging of EVs could improve the synergy between photovoltaic, EVs, and electricity consumption, leading to both technical and economic advantages. Reference [17] presents a reputation-based framework for allocating power to plug-in EVs in the smart grid. In this framework, the available capacity of the distribution network measured by distribution-level phasor measurement units is divided in a proportionally fair manner among connected EVs, considering their demands and self-declared deadlines. In [18] main aspects of smart charging reviewed are objectives, configurations, algorithms, and mathematical models, and the commonly employed optimization techniques and rule-based algorithms for smart charging are reviewed.

With the growing concerns about energy depletion and the reduction of CO_2 emissions, EVs have gained popularity in the transport sector due to clean and reliable energy sources. Reference [19] aims to investigate the optimal EV coordination with the V2G technology for the cost-benefit analysis. Battery degradation cost is formulated for real-time analysis taking the depth of discharge at each time interval. The firefly algorithm has been used to optimize the system cost.

Proper accommodation of EVs poses significant challenges to distribution system planning and operations. In [20] two scheduling strategies are implemented considering active power dispatch and reactive power dispatch from the EVs. The objective of both strategies is to minimize losses in the system by utilizing the V_2G operation of the EVs. In [21], the authors investigate the achievement of energy management strategies in the EV system, which reduces fuel consumption and carbon dioxide emissions. The novelty of this article is an update on the most advanced technology in the field of V_2G and energy management strategy.

Many researchers try to optimize the charging and discharging pattern of EVs by using a centralized approach and considering a series of predetermined criteria.

Simultaneous consideration of the effects of distributed generation, especially of the renewable type and types of EVs, is another important issue in the studies of distribution networks. This point affects the future power systems of many countries. In this regard, some of the studies carried out are given below.

When both renewable energy sources generation and utilization occur simultaneously, energy storage costs can be reduced, and voltage oscillation and system instability caused by renewable energy sources grid connection can be reduced. Reference [22] constructs a microgrid model that includes EVs, defines the charge and discharge capacity of EVs, and uses the flexibility of EVs to overcome the intermittency and volatility of renewable energy sources.

Managing uncertainty is key to enhancing robustness in microgrids. In [23], the authors focus on the uncertainties in aggregated EVs and establish a two-layer model predictive control strategy for charging EVs with a microgrid. The main concern related to renewable-based distributed generators, especially photovoltaic and wind turbine generators, is the continuous variations in their output powers due to variations in solar irradiance and wind speed, which leads to uncertainties in the power system. Reference [24] proposes an efficient stochastic framework for the optimal planning of distribution systems with optimal inclusion of renewable-based distributed generators, considering the uncertainties of load demands and the output powers of the distributed generators.

The researchers in reference [25] stated that the simultaneous presence of all types of EVs as well as a large number of wind turbines with low capacity has created many technical challenges for distribution network operators to provide reliable energy and optimize energy distribution. Three different approaches for the simultaneous distribution of these small generation sources (wind turbines) and potential distributed reserves (EVs) are introduced. The electricity in a microgrid is such that this energy is provided in time intervals with low load. For this purpose, a condition has been added that it cannot be interrupted when the vehicle charging process starts. The results of applying this method to the sample network show give that this method with this stipulation cannot direct a load of vehicles in a direction that is most compatible with the generation power of wind turbines. In the second method used under the title of interruptible distribution, the same goal as the previous method is used. The vector is followed with the difference that it can be in this approach the process of charging vehicles intermittently in studies. The third method, called the spread method with different charging rates, like the previous two methods, tries to optimize the charging process of vehicles to have the most agreement with the generation graph of wind turbines, with the difference that it considers different charging rates for different vehicles.

Reference [26] describes a two-level planning model. In the proposed model, resource planning has been done at the two levels of aggregators and system operators. In the first stage, after the aggregators have received all the information related to vehicles and distributed renewable generation sources, they inform the operator of the amount of energy they need or their excess energy by performing calculations. In the second stage, the system operator plans energy generation and storage to reduce costs. The results of this research show that with the proper management of EVs, the electric load of the network does not increase sharply during peak hours, in other words, the use of the proposed planning model has flattened the network load curve.

Studies show that less research has been done on the simultaneous management of distributed generation resources and EVs as two completely independent and private entities, and on the other hand, the owners of distributed generation resources and EVs are looking for maximum profit, this factor can cause problems such as increasing losses, line congestion, increasing network strengthening costs, etc. in distribution networks. Therefore, in this article, by presenting a two-stage planning framework, EVs and dispersed generation resources with private ownership, whose goal is to maximize their respective profits, will be managed in such a way that in addition to their high satisfaction This important issue, i.e. reduction of operating costs, should be addressed by considering network limitations.

The innovative aspects of the article are described as follows:

- A two-stage planning framework for managing energy resources in a distribution network is presented to achieve an application and take into account the demands and needs of different agents.
- The optimization problem related to the planning of charging and discharging of EVs has been modeled from the point of view of their owners and considering their uncertainty.
- The optimization problem related to the planning of distributed generation resources has been modeled and solved, and its effect has been included in the planning problem of energy resources of the distribution network.
- The problem of optimizing the use of energy resources is linearized.

The article is organized in such a way that in the second part of the problem statement, the proposed planning framework and formulation of the problem are discussed. The case study is described in the third section and finally, the results are presented in the fourth section.

2. Materials and Methods

2.1. Statements of the problem

In this part, the modeling process of energy planning in the distribution network with the proposed method is described. First, the planning framework of the proposed model is described. Then the formulation of the problem is given along with the planning constraints.

Proposed energy planning framework

Due to the increase in the presence of EVs and distributed sources of generation in distribution networks, the need for a suitable control program to control the process of charging and discharging vehicles as a new load and sources of distributed generation as a source of energy generation is felt more and more [27, 28]. In the following, a two-stage algorithm is presented to achieve a comprehensive planning framework in which not only the technical constraints of the network are met, but also the privacy and convenience of EV owners, distributed generation resources, and other actors are considered. The proposed planning framework consists of two stages.

In the first stage, the coordinators of EVs and the owners of distributed generation resources try to maximize their profit during the planning period by implementing a separate optimization program, taking into account their demands and limitations. For this purpose, the owners of EVs provide the coordinators with information such as the time to arrive at the parking lot, the time to leave the parking lot, the initial charging status, and the final charging status, so that the optimal charging/discharging program for the vehicles can be obtained; And on the other hand, the owners of distributed generation resources try to maximize their profits by having information about distributed generation resources and electricity market prices. After the end of the first stage of the proposed planning, the optimal charging/discharging program related to the vehicles and the generation pattern of the units will be reported to the network operator.

In the second stage of the proposed energy planning, after receiving the optimal charge/discharge plan for vehicles and the plan for the generation of distributed generation sources, in each scenario, by purchasing energy from the market, the network operator tries to change the optimal generation plan of the distributed generation sources and change The optimal vehicle charging/discharging program will plan the energy of the available resources in such a way as to reduce the operating costs while providing the required load of the network. The resource usage pattern, EV charging/discharging schedule along with the power purchased from the grid are the primary outputs of this planning stage. Further, although these outputs are optimal from the point of view of vehicle owners and distributed generation resources, they do not provide any guarantee regarding the technical limitations of the network. Therefore, the network operator checks all the technical restrictions of the network after carrying out the load distribution calculations and if any of the restrictions are not met, he repeats the second stage of optimization by applying new restrictions. This work continues until all network constraints are met. Below is the formulation for each step.

2.2. Formulation of the problem

Formulation of the first stage of the proposed planning

In the first stage of the planning framework, coordinators and owners of distributed generation resources seek to maximize their profits by implementing optimization problems, below are the relationships related to each.

Formulation related to the coordinator of vehicles

The objective function related to maximizing the profit of vehicles is calculated from Equation (1).

$$f_{1,1} = max\left(\sum_{t=1}^{T}\left[\sum_{v=1}^{V}\begin{cases}P_{EV}^{Dcharge}(v,t)\times Pr_{EV}^{Dcharge}(t\\ -P_{EV}^{Charge}(v,t)\times Pr_{EV}^{Charge}(t\end{cases}\right.\right. \times \Delta t\Bigg) \quad (1)$$

The restrictions related to vehicles are as follows [26]:

In planning the charging and discharging of a vehicle, it should be noted that the vehicle should not be programmed in two charging and discharging modes at the same time.

$$X(v,t)+Y(v,t)\le 1\ \forall t\in\{1,2,\dots,T\};\ \forall v\in\{1,2,\dots,V\}; X,Y\in\{0,1\} \quad (2)$$

The time continuity equation of vehicle charging and discharging during the planning period is given as the following relationship.

$$E_s(v,t)=E_s(v,t-1)+\eta_v^{Charge}\times P_{EV}^{Charge}(v,t)\times\Delta t-\frac{1}{\eta_v^{Dcharge}}\times\left(P_{EV}^{Dcharge}(v,t)\times\Delta t\right)\quad \forall t\in\{1,2,\dots,T\};\ \forall v\in\{1,2,\dots,V\} \quad (3)$$

The limit of chargeable power and battery discharge of each vehicle in each period are as follows:

$$P_{EV}^{Charge}(v,t)\le P_{Charge,v}^{Max}\times X(v,t)\ \forall t\in\{1,2,\dots,T\};\ \forall v\in\{1,2,\dots,V\} \quad (4)$$

$$P_{EV}^{Dcharge}(v,t)\le P_{Dcharge,v}^{Max}\times Y(v,t)\ \forall t\in\{1,2,\dots,T\};\ \forall v\in\{1,2,\dots,V\} \quad (5)$$

Discharging the battery of an EV up to a certain maximum value ψ_v^{Min} and charging it up to a certain maximum value ψ_v^{Max} will prevent the premature breakdown of the battery and increase its useful life [26].

$$E_s(v,t) \le \psi_v^{Max}\ \forall t \in \{1,2,\dots,T\};\ \forall v \in \{1,2,\dots,V\} \tag{6}$$

$$E_s(v,t) \ge \psi_v^{Min}\ \forall t \in \{1,2,\dots,T\};\ \forall v \in \{1,2,\dots,V\} \tag{7}$$

Where ψ_v^{Min} and ψ_v^{Max} are calculated as follows:

$$\psi_v^{Max} = \varphi_v^{Max} \times E_{BatCap,v}\ \ \forall v \in \{1,2,\dots,V\} \tag{8}$$

$$\psi_v^{Min} = \varphi_v^{Min} \times E_{BatCap,v}\ \ \forall v \in \{1,2,\dots,V\} \tag{9}$$

The limitation of charging and discharging the battery every hour is applied according to the amount of energy stored in the battery in the previous period and the maximum capacity of the battery [26]:

$$\frac{1}{\eta_v^{Dcharge}} \times \left(P_{EV}^{Dcharge}(v,t) \times \Delta t\right) \le E_s(v,t-1)\ \forall t \in \{1,2,\dots,T\};\ \forall v \in \{1,2,\dots,V\} \tag{10}$$

$$\eta_v^{Charge} \times P_{EV}^{Charge}(v,t) \times \Delta t \le \left(\psi_v^{Max} - E_s(v,t-1)\right)\ \forall t \in \{1,2,\dots,T\};\ \forall v \in \{1,2,\dots,V\} \tag{11}$$

The optimal amount of stored energy in the battery of each vehicle at the time of leaving the parking lot is given in the following equation:

$$SOC_{des}^{v} = SOC_{initial}^{v} + randnumber(0,[1 - SOC_{initial}^{v}])\forall v \in \{1,2,\dots,V\} \tag{12}$$

The limit of the number of times the status changes from charging to discharging and vice versa according to the life of the vehicle battery is given in the following equation [28]:

$$D^v \le NS^{MAX} \tag{13}$$

By performing linear programming with binary variables, the desired charge/discharge profile of vehicles is obtained as follows.

$$\mathrm{P}_{Des}^{Charge}(v) = \left[P_{EV}^{Charge}(v,t)\right]\ v \in [1-V], t \in \{1-T\} \tag{14}$$

$$\mathrm{P}_{Des}^{Dcharge}(v) = \left[P_{EV}^{Dcharge}(v,t)\right]\ v \in [1-V], t \in \{1-T\} \tag{15}$$

Formulation related to distributed non-renewable generation sources

Since non-renewable distributed generation resources are considered to be privately owned, the objective function related to them (maximizing profit) is in the form of Equation (16).

$$f_{1,2} = max\left(\sum_{t=1}^{T}\left[\sum_{j=1}^{J}\{P_{DG}(j,t) \times Pr_{MRT}(t) - C_{DG}(j,t)\}\right] \times \Delta t\right) \tag{16}$$

The restrictions related to distributed non-renewable generation sources are as follows:

The cost of non-renewable resources is modeled as a function of their output power. To use the optimization method of linear programming, the cost functions with a suitable approximation are considered as follows [28].

$$C_{DG}(j,t) = a_j + b_j \times P_{DG}(j,t)\ \forall t \in \{1,2,\dots,T\};\ \forall j \in \{1,2,\dots,J\} \tag{17}$$

Limits on the maximum and minimum generation capacity of distributed non-renewable generators are in the form of the following equations:

$$P_{DG}(j,t) \le P_{DG,j}^{Max} \times u(j,t)\ \forall t \in \{1,2,\dots,T\};\ \forall j \in \{1,2,\dots,J\} \tag{18}$$

$$P_{DG}(j,t) \ge P_{DG,j}^{Min} \times u(j,t)\ \forall t \in \{1,2,\dots,T\};\ \forall j \in \{1,2,\dots,J\} \tag{19}$$

The cost of setting up non-renewable distributed generation generators is calculated as follows [28]:

$$SU(j,t) = Sc_j \times (u(j,t) - u(j,t-1)) \tag{20}$$

$$SU(j,t) \ge 0 \tag{21}$$

The limit of the rate of increase and decrease of power related to non-renewable distributed generation sources is as follows:

$$(P_{DG}(j,t+1) - P_{DG}(j,t)) \le RUP_{DG}^{j} \tag{22}$$

$$(P_{DG}(j,t) - P_{DG}(j,t+1)) \le RDN_{DG}^{j} \tag{23}$$

By performing linear programming with binary variables, the optimal generation pattern of distributed generation resources is obtained as follows.

$$\mathrm{P}_{Des}^{DG}(j) = [P_{DG}(j,t)]\ j \in [1-J], t \in \{1-T\} \tag{24}$$

Formulation of the second stage of the proposed planning

In the second stage, the network operator, after receiving the information of the first stage (14, 15, and 16 equations) in each scenario, tries to change the optimal generation plan of distributed generation resources and also change the optimal charge/discharge profile of vehicles, by purchasing energy from the market, planning energy resources to do the existing in such a way as to guarantee the supply of EV owners and distributed generation resources, reduce the network's technical limitations and operating costs. To achieve these goals, the following optimization program is performed by the system operator for all scenarios:

$$f_2 = min\left(\sum_{t=1}^{T}\left[P_{NTW}(t) \times Pr_{MRT}(t) + P_{LOSS}(t) \times Pr_{LOSS}(t) + \left|\sum_{j=1}^{J}\{P_{DG}(j,t) - P_{Des}^{DG}(j,t)\}\right| \times K_{DG} + \left|\sum_{v=1}^{V}\{P_{EV}^{Charge}(v,t) - P_{Des}^{Charge}(v,t)\}\right| \times K_{Charge} + \left|\sum_{v=1}^{V}\{P_{EV}^{Dcharge}(v,t) - P_{Des}^{Dcharge}(v,t)\}\right| \times K_{Dcharge}\right] \times \Delta t\right) \quad (25)$$

As it is clear from Equation (25), the objective function of the optimization program at this stage includes four parts. The first part shows the cost of energy purchased from the market and the cost of losses. The cost paid to owners of distributed generation resources and EV owners Electric to participate in the proposed program is given in the second, third, and fourth parts.

2.3. Network restrictions

Adverb of power balance

The total power produced along with the power purchased from the flash market should be equal to the amount of consumption.

$$\mathrm{P}_{NTW}(t) + \sum_{w=1}^{W} P_w(t) + \sum_{pv=1}^{PV} P_{pv}(t) + \sum_{j=1}^{J} P_{DG}(j,t) + \sum_{v=1}^{V} P_{EV}^{Dcharge}(v,t) = \sum_{v=1}^{V} P_{EV}^{Charge}(v,t) + P_{LOAD}(v,t) + P_{LOSS}(t)\ \forall t \in \{1,2,\dots,T\} \quad (26)$$

Network technical restrictions

The technical limitations related to the network are given below [21]:

$$P_n(t) = \sum_{m=1}^{N} |V_n(t)||V_m(t)||Y_{n,m}(t)| \cos(\delta_m(t) - \delta_n(t) + \theta_{n,m})\ \forall n,t \quad (27)$$

$$Q_n(t) = -\sum_{m=1}^{N} |V_n(t)||V_m(t)||Y_{n,m}(t)| \sin(\delta_m(t) - \delta_n(t) + \theta_{n,m})\ \forall n,t \quad (28)$$

$$|S(n,m,t)| \le S_{n,m}^{max}\ \forall t \in \{1,2,\dots,T\};\ \forall n,m \in \{1,2,\dots,N\} \quad (29)$$

$$V_n^{min} \le V(n,t) \le V_n^{max}\ \forall t \in \{1,2,\dots,T\};\ \forall n \in \{1,2,\dots,N\} \quad (30)$$

$$P_{NTW}(t) \le P_{NTW}^{max}\ \forall t \in \{1,2,\dots,24\} \quad (31)$$

$$P_{TRANS}(n,t) \le P_{TRANS}^{max}\ \forall t \in \{1,2,\dots,T\};\ \forall n \in \{1,2,\dots,N\} \quad (32)$$

Restrictions related to EVs

Equations 2-13 are constraints related to EVs that should be considered in this phase of energy planning.

2.4. Restrictions related to distributed generation sources

Non-renewable resources

Equations 18-23 are constraints related to distributed non-renewable generation sources that must be considered in energy planning.

Renewable resources

Since the primary energy source of wind turbines and photovoltaic units is the wind and the sun, in the existing studies, probabilistic functions are used to model their output power, which will be described below.

Probability model of the photovoltaic system

In this study, the beta probability density function is used to model the power of the photovoltaic system [29].

$$f(I_r^t) = \begin{cases} \dfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \times I_r^{t(\alpha-1)} \times (1 - I_r^t)^{\beta-1} \\ for\ 0 \le I_r^t \le 1, \alpha \ge 0, \beta \ge 0 \\ 0 \qquad otherwise \end{cases} \quad (33)$$

According to the radiation intensity distribution predicted in each area and the radiation-to-power conversion function, the output power of the photovoltaic system can be calculated for each radiation intensity at any time [30].

$$P_{pv} = \eta^{pv} \times S_r^{pv} \times I_r^t(1 - 0.005 \times (T_a - 25)) \quad (34)$$

Wind turbine probabilistic model

In this study, Rayleigh's probability density function is used to model wind speed behavior [31].

$$f(v_f^t) = \left(k/_c\right) \times \left(v_f^t/_c\right)^{(k-1)} e^{-\left(v_f^t/_c\right)^k}\ 0 \le v_f^t \le \infty \quad (35)$$

Also, the output power of the wind turbine at any moment can be calculated using the power conversion function given in the following relationship [32].

$$P_w = \begin{cases} 0 & 0 \le v_f^t \le v_{ci} \\ \mathrm{P}_{rated} \times \dfrac{(v_f^t - v_{ci})}{(v_r - v_{ci})} & v_{ci} \le v_f^t \le v_r \\ \mathrm{P}_{rated} & v_r \le v_f^t \le v_{co} \\ 0 & v_{co} \le v_f^t \end{cases} \quad (36)$$

Figure 1 shows the flowchart of the proposed energy planning. As it is clear, the coordinators and owners of distributed generation resources have obtained the charging/discharging profile of vehicles and the generation pattern of distributed generation resources by implementing the optimization program. Then, for all scenarios, the operator should implement non-linear programming with binary variables (equation 25) of the output power corresponding to each of the distributed generation sources, the power purchased from the network, and the charging correction strategy to determine the discharge related to the number of vehicles. Since the second stage of optimization has non-linear terms (the absolute value terms in equation 25), there is no guarantee to extract the absolute optimal solution. Therefore, at first, these relations are sub-linearized [33].

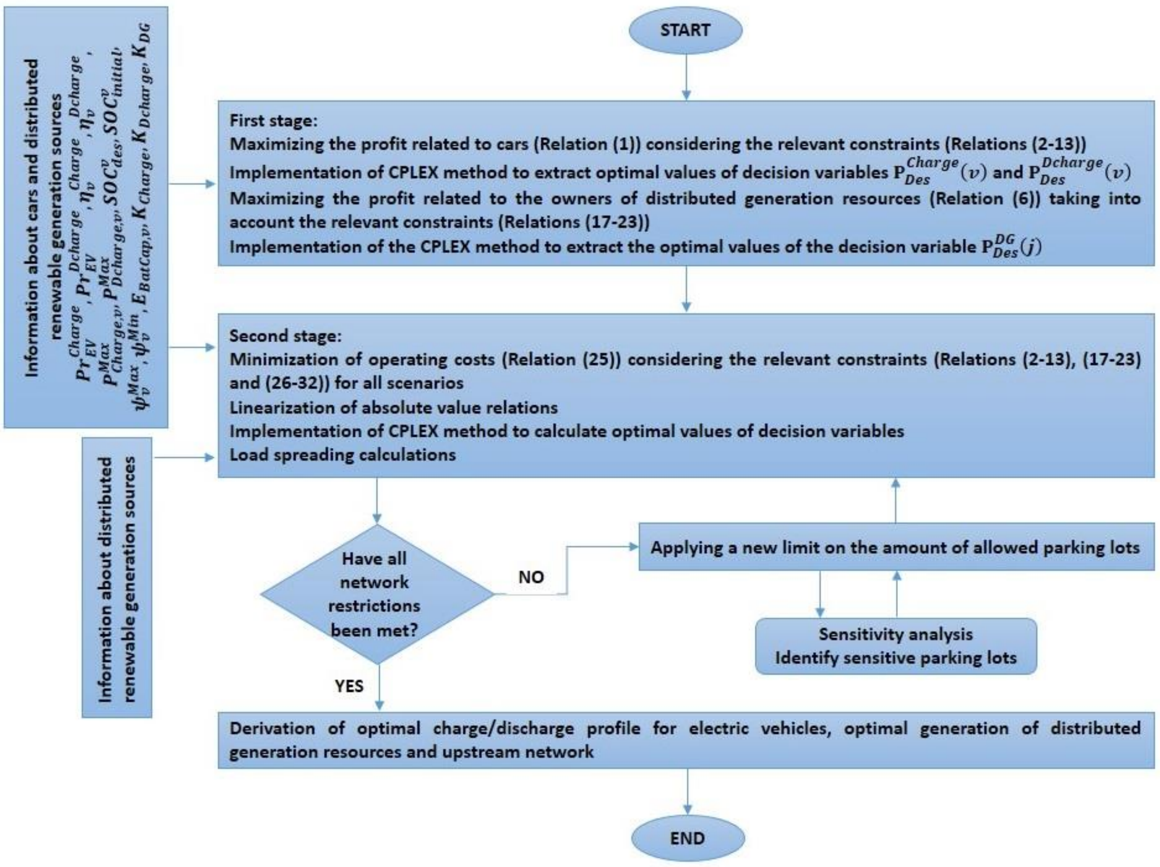

Information about cars and distributed renewable generation sources
$Pr_{EV}^{Charge}, Pr_{EV}^{Dcharge}, \eta_v^{Charge}, \eta_v^{Dcharge},$
$P_{Charge,v}^{Max}, P_{Dcharge,v}^{Max}, SOC_{des}^v, SOC_{initial}^v,$
$\psi_v^{Max}, \psi_v^{Min}, E_{BatCap,v}, K_{Charge}, K_{Dcharge}, K_{DG}$
Information about distributed renewable generation sources
START
First stage:
Maximizing the profit related to cars (Relation (1)) considering the relevant constraints (Relations (2-13))
Implementation of CPLEX method to extract optimal values of decision variables $P_{Des}^{Charge}(v)$ and $P_{Des}^{Dcharge}(v)$
Maximizing the profit related to the owners of distributed generation resources (Relation (6)) taking into account the relevant constraints (Relations (17-23))
Implementation of the CPLEX method to extract the optimal values of the decision variable $P_{Des}^{DG}(j)$
Second stage:
Minimization of operating costs (Relation (25)) considering the relevant constraints (Relations (2-13), (17-23) and (26-32)) for all scenarios
Linearization of absolute value relations
Implementation of CPLEX method to calculate optimal values of decision variables
Load spreading calculations
Have all network restrictions been met?
NO
Applying a new limit on the amount of allowed parking lots
Sensitivity analysis
Identify sensitive parking lots
YES
Derivation of optimal charge/discharge profile for electric vehicles, optimal generation of distributed generation resources and upstream network
END

Figure 1. Proposed energy planning flowchart.

Assuming that two variables ε and γ are positive:

$$minimize|f(x)| \to minimize\ \gamma + \varepsilon$$
$$f(x) = \gamma - \varepsilon$$
$$\gamma, \varepsilon \ge 0$$

Finally, the network operator checks the technical limitations of the network in Equations (29-32) by implementing the load distribution. If any of the restrictions are not met, the second stage of planning is repeated by applying restrictions on the amount of load related to sensitive parking lots (sensitive tires) until the restrictions are fully met. It should be noted that sensitivity analysis was used to determine the sensitive parking lots, that is, for each period, for each parking lot, the load of the parking lot increased by 10%, and the changes related to the voltage of the parking lots were saved. After applying this algorithm, sensitive parking lots are identified each period.

2.5. Case studies

Introduction of the studied system

The proposed planning framework has been tested on a distribution network connected to bus 5 of the

RBTS sample network, which has 4 feeders at a voltage of 20 kV [34]. For this network, the data related to the type and number of subscribers connected to different load points, the average load of each of them are presented in table 1. This network along with the division of the areas related to the coordinators is shown in Figure 2. The voltage limit of the bus bars is considered equal to 0.9-1.05 per unit.

Table 1. The type and average amount of load and the number of subscribers of different load points in the distribution network under study.

Number of subscribers	Average load (MW)	Subscriber type	Load points
180	0.4569	Residential	1-2-20-21
2	0.6646	Official	3-5-8-17-23
250	0.4771	Residential	4-6-15-25
2	0.4089	Commercial	7-14-18-22-24
210	0.2513	Residential	9-10-11-13-26
2	0.4086	Official	12-16-19

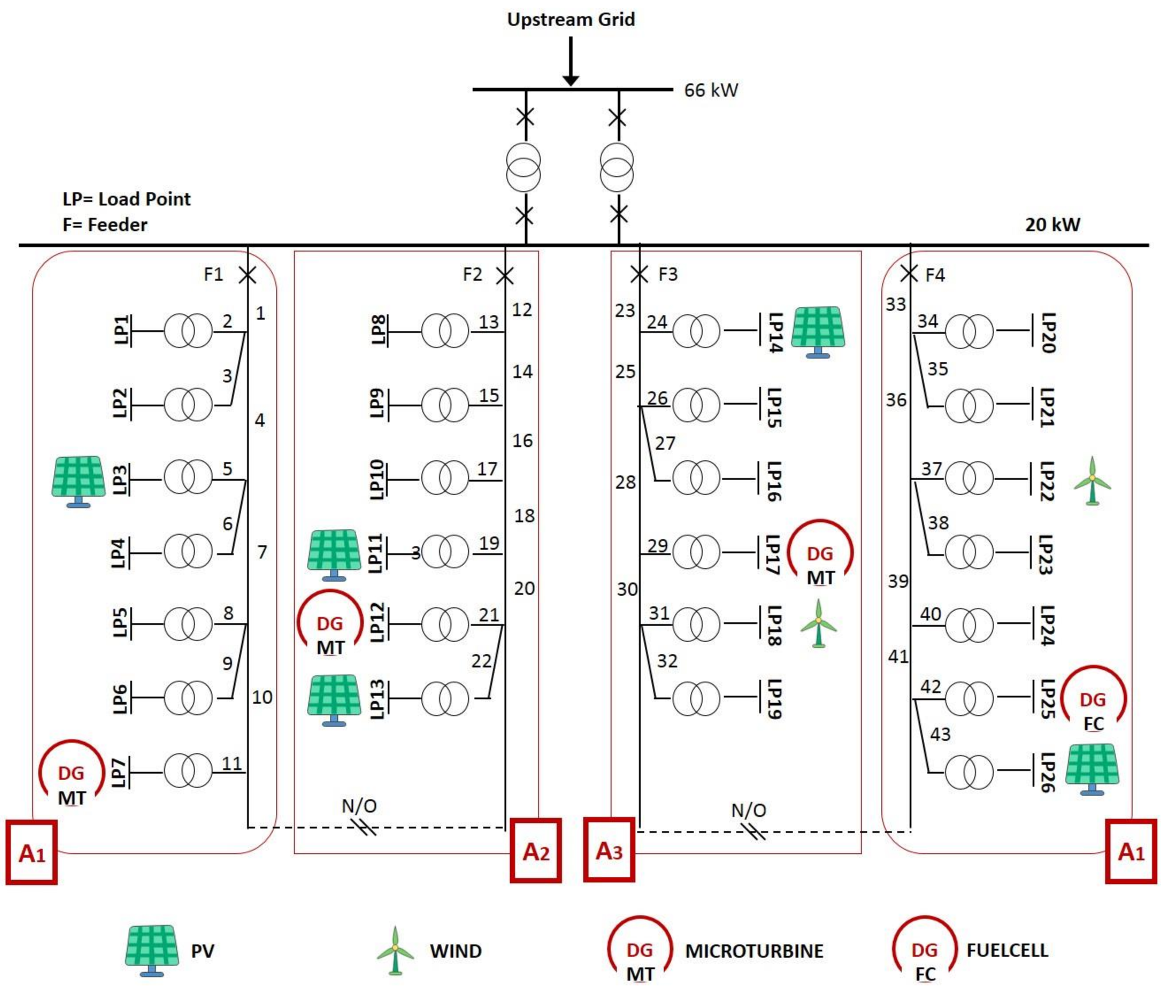


Figure 2. One-line diagram of the distribution network connected to bus 5 of the RBTS sample network.

The hourly price of the electricity market is given in table 2 [35]. The capacity of medium-pressure and low-pressure transformers of the network is considered to be MVA 1 and MVA 15 respectively. In this network, there are four coordinators named A1, A2, A3, and A4 are considered. The predicted hourly load of each of the coordinators in the 24-hour planning period is shown in Figure 3.

Table 2. Hourly electricity market price.

Hour	Price ($/kWh)	Hour	Price ($/kWh)
1	0.033	13	0.215
2	0.027	14	0.572
3	0.02	15	0.286
4	0.017	16	0.279
5	0.017	17	0.086
6	0.029	18	0.059
7	0.033	19	0.05
8	0.054	20	0.061
9	0.215	21	0.181
10	0.572	22	0.077
11	0.572	23	0.043
12	0.572	24	0.037

Also, three microturbine units and one fuel cell unit have been installed in this network. The specifications of the cost functions of each of the units are given in table 3. The maximum rate of increase and decrease of power related to each of the units in each period is equal to 20% of their maximum capacity.

Table 3. Characteristics of distributed non-renewable generation units.

Gen type	a ($)	b ($/kW)	P_{DG}^{Min} (kW)	P_{DG}^{Max} (kW)
MT	20	0.2	50	350
MT	40	0.3	50	250
MT	20	0.2	50	350
FC	90	0.35	50	250

The pattern of using vehicles has been obtained according to a statistical study in the city of Tehran. The obtained information includes the entry and exit times of the vehicles, the amount of initial energy when entering the parking lots, and other information related to the vehicles. A summary of information on the behavior pattern of vehicle owners in using their vehicles is given in Table 4.

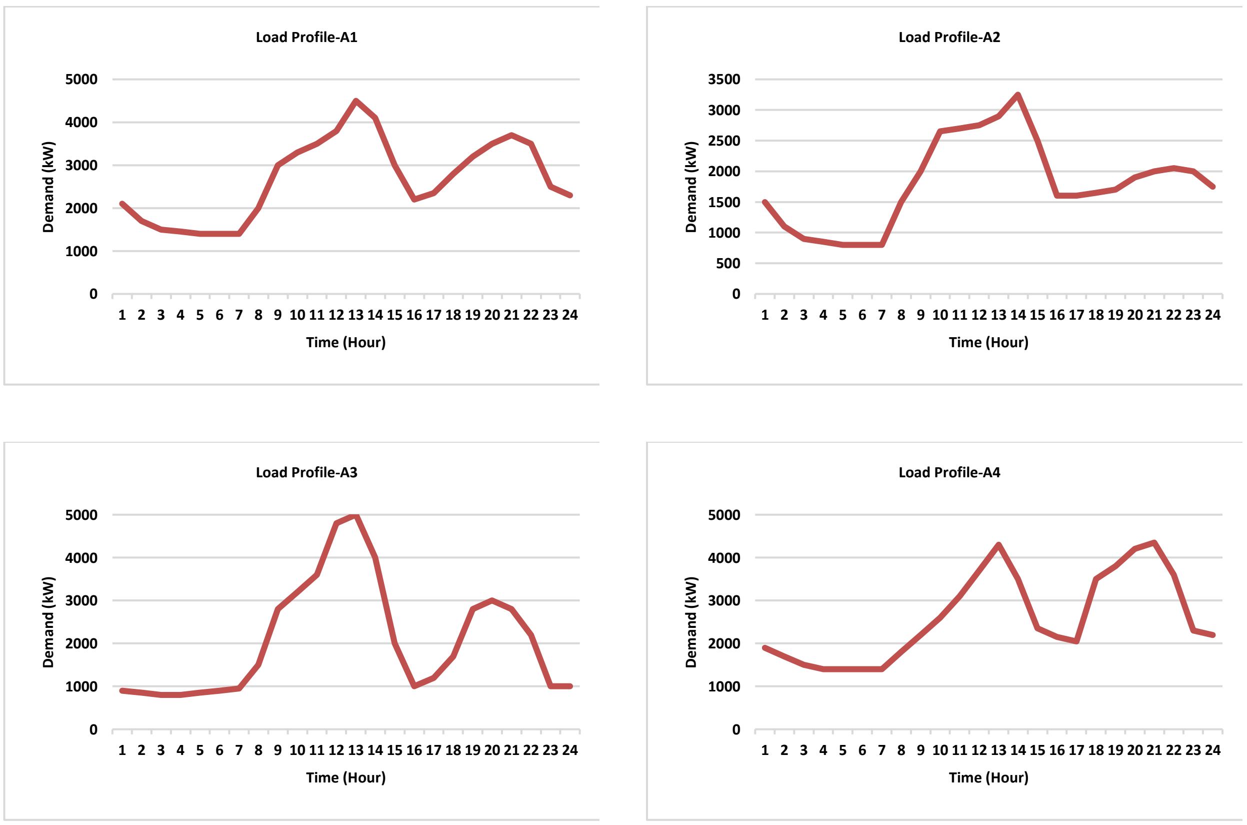


Figure 3. Hourly load demand of coordinators.

Table 4. Statistical information on EVs.

Type	Arrival time (H)	Departure time (H)	
Residential	Norm (19, 5)	Norm (7, 2)	I [0.1, 0.5]
Official	Norm (7, 1)	Norm (15, 1)	I [0.5, 0.8]
Commercial	Norm (9, 2)	Norm (20, 2)	I [0.3, 0.6]

To calculate the total number of vehicles, the first step is to know the number of residential subscribers covered by the network. In this regard, the information presented in table 1 was used and finally, for 35% penetration, the total number of vehicles in the network was estimated to be 4004 vehicles. To conduct studies, in addition to the number of vehicles, their class is also according to [36] considered.

Battery capacity is one of the important features of vehicles. According to [36], the range of battery capacity in each class is considered in Table 5. It should be noted that a uniform distribution has been used to distribute the capacity of batteries in each class.

Table 5. Battery capacity range for each class.

Class	Minimum capacity (kWh)	Maximum capacity (kWh)
1	8	12
2	10	14
3	17	21
4	19	24

The maximum charging and discharging rate of vehicles is 4 kWh and the weighted coefficients of charging and discharging are equal to 60% of the market peak price. Usually, some energy is lost in the process of charging and discharging vehicle batteries, therefore, the efficiency coefficient of charging and discharging vehicles is considered to be 90% and 95% [26]. Also, to prevent premature aging of vehicle batteries, battery discharge is allowed up to 85%, and the number of switching times allowed is considered according to table 6. It should be noted for the vehicles under study; their battery life is randomly selected.

Table 6. The number of times allowed to replace vehicle batteries according to their lifespan.

AOB<4	4<=AOB<6	6<=AOB<8	8<=AOB	Battery life
8	6	4	2	NS^{Max}

In this study, it is assumed that all the wind turbines installed in the network are of the same model and their specifications are according to table 7 [28].

Table 7. Information about wind turbines.

Vco (m/s)	Vr (m/s)	Vci (m/s)	Prated (kW)
30	12	3	500

Also, photovoltaic systems with a power of 100 kW (10 panels of 10 kW) have been installed at the network level, whose specifications are given in Table 8 [26]. In all studies, it is assumed that the photovoltaic system and wind turbines are operated at the unit power factor.

Table 8. Photovoltaic system information.

η (%)	S(m^2)	Ta (ºC)
18.6	10	25

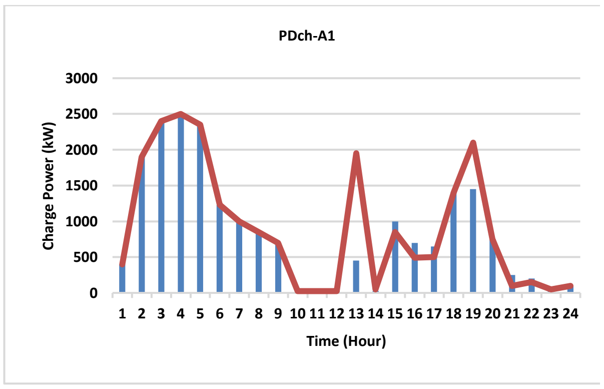
To generate scenarios for each planning interval, the distribution function of wind speed and solar radiation is divided into five intervals, so that these functions are converted from continuous to discrete. To reduce the execution time and the complexity of the program, the number of scenarios was first reduced with the help of the backward scenario reduction technique [37] and then the wind and solar power scenarios were combined to obtain the final scenarios. In this study, the number of final scenarios is 10.

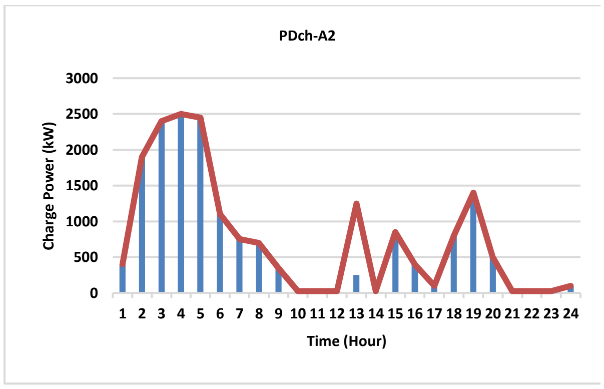
Simulation process

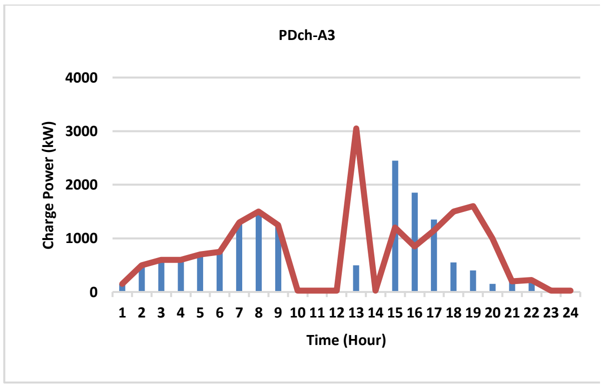
The proposed programming method is coded in OpenDSS, GAMS, and MATLAB software. The first and second stages of this planning are linearly implemented and the CPLEX calculation method is used to solve the problem.

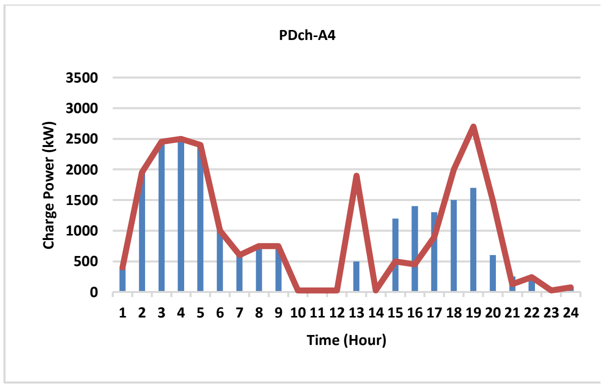
3. Results and Discussion

The results of the planning done in both stages of the proposed planning are shown in Figures 4 and 5. The results show that the charging of most of the vehicles took place during the low load hours of the network (1-7 in the morning) because, during these hours, the price of the electricity market is low. Also, the discharge of vehicles during peak hours of the network has reduced the peak load of the network and met the technical limitations of the network. It should be noted that according to figures 4 and 5, the highest cost paid to vehicles in the direction of participation was during peak hours. Figure 6 shows the planned power of distributed non-renewable generation resources for two stages of the proposed algorithm.



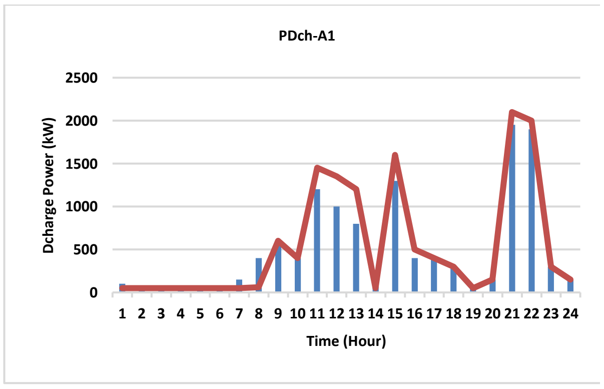


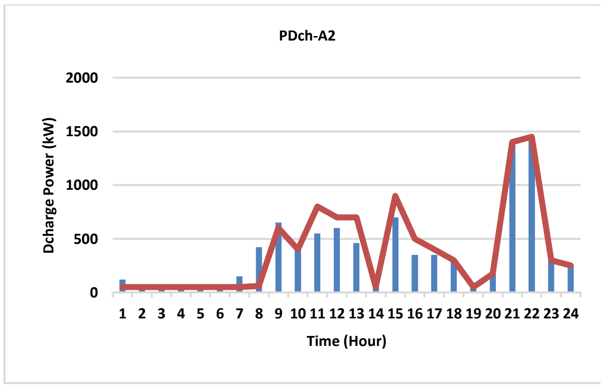




Stage 1 ▬ **Stage 2** 🟦

Figure 4: Charging profile of EVs in different areas after applying the proposed two-stage algorithm





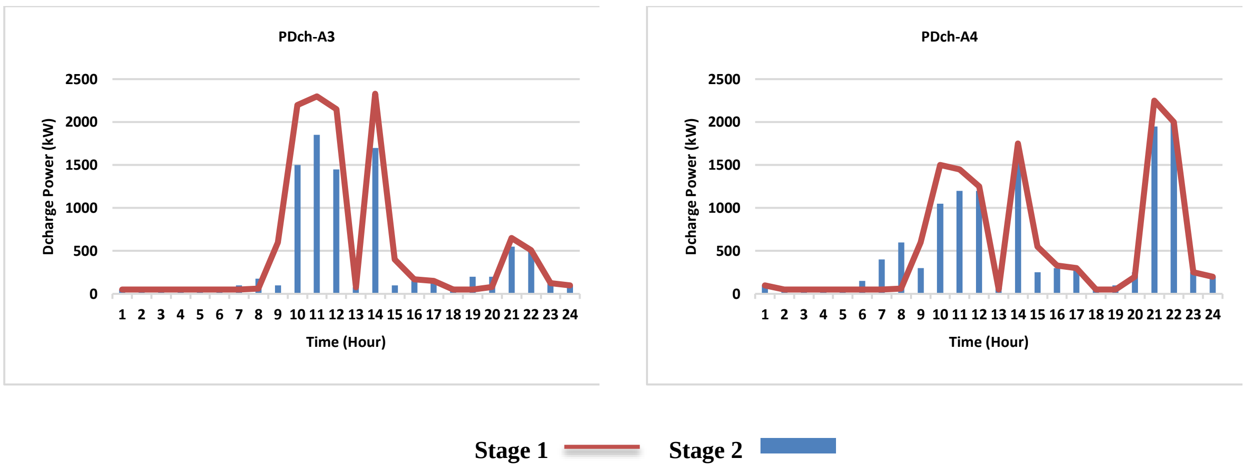


Figure 5. Discharge profile of EVs in different areas after applying the proposed two-stage algorithm.

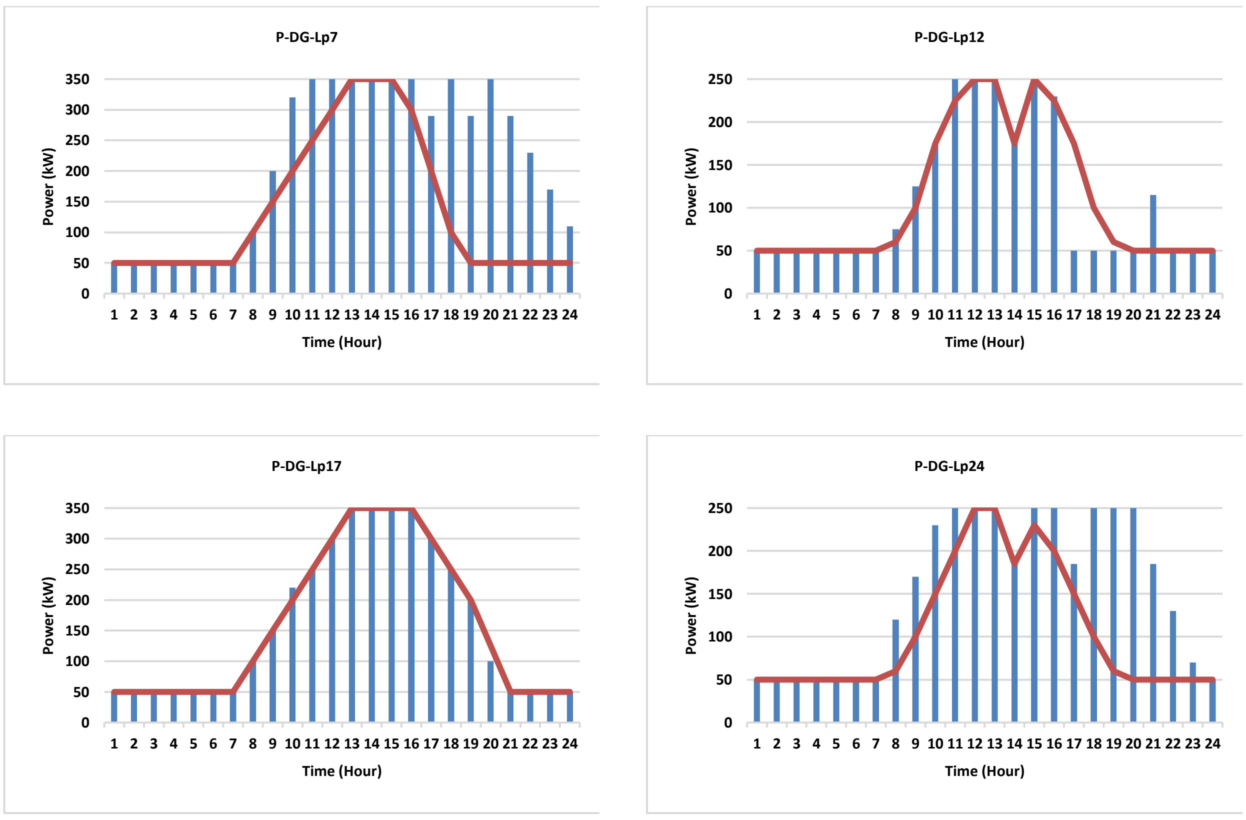


Figure 6. The generation capacity of each non-renewable distributed generation source (first stage-second stage).

(The dashed line curve represents the first stage of the proposed algorithm and the load curve corresponds to the second stage of the proposed algorithm). As it is known, non-renewable resources produce their maximum power during the peak hours of the network due to the high price of the energy market and in Low load hours due to the low price of the electricity market, the minimum power is produced and the available load is supplied by the upstream network.

Figure 7 shows the network load profile for three different network operation situations. As it is clear, the presence of EVs as a new load has changed the shape of the network load profile (gray color curve - vehicles cause an increase of about 6 megawatts have become the peak load of the network. The application of the proposed planning framework has caused the electric load of the network to not increase much during peak hours (dashed line curve). The load has increased, which has made the network load profile curve more uniform.

Further, to check the efficiency of the proposed algorithm, studies have been carried out for cases where vehicles and distributed generation sources do not participate in the proposed plan. According to the obtained results, it was found that the existing network is not sufficient to provide the load required by the vehicles (the peak load of the network has increased by almost 7 megawatts) and there is a need to strengthen the network.

Therefore, for comparison, the costs related to one year of implementation of the proposed plan were compared with the costs related to the network strengthening plan.

The network strengthening plan has been carried out in such a way that the studies related to the network strengthening plan with 35% penetration of vehicles for a 20-year horizon were carried out and the related costs were obtained. Finally, for comparison, the costs of strengthening the network were obtained by considering the interest rate of 10% for one year of equalization. Table 9 shows the costs obtained after the implementation of the two plans, as it is clear that the increase in the costs of strengthening the network is more than 5 times the increase in the costs of implementing the proposed plan.

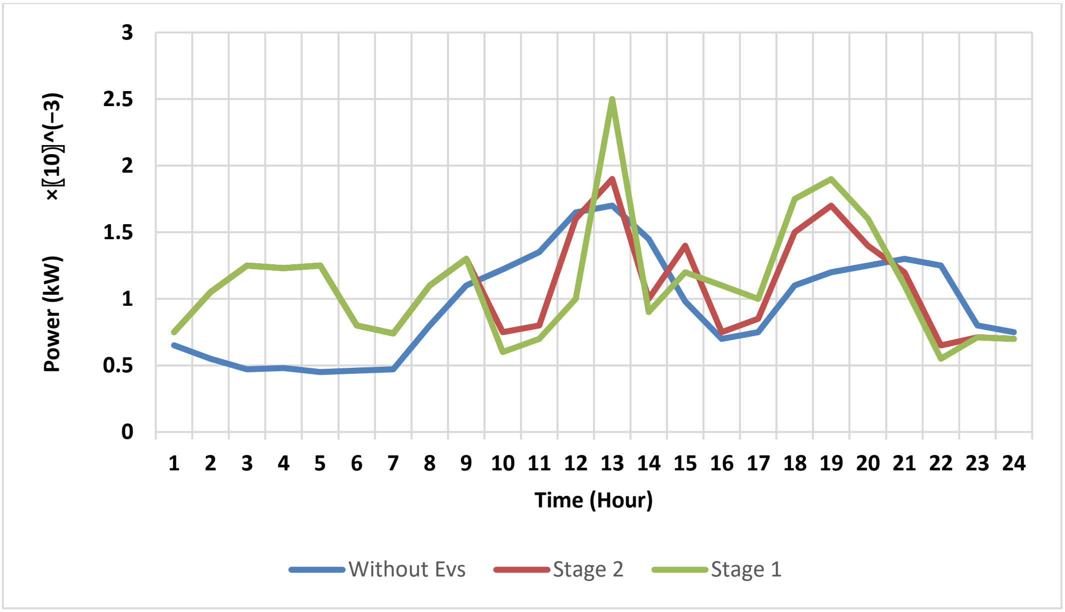


Figure 7. Network load profiles for three different operating states.

Table 9. Increase in costs.

Plan	Increase in the annual cost of implementing the proposed plan ($)
Proposed plan	1601941
Network strengthening plan	9058207

4. Conclusions

In this article, a two-stage planning framework for the optimal management of EVs and dispersed generation resources with private ownership was analyzed in a centralized manner. The basic approach in this article was the optimal charging/discharging planning of EVs along with distributed generation sources to reduce operating costs by considering the wishes of vehicle owners and distributed generation sources. To implement the proposed algorithm, at first EVs were modeled probabilistically and the uncertainty related to distributed renewable generation sources was considered, then the CPLEX optimization method was used for different scenarios to solve the problem. Finally, as the results show, the use of the proposed planning model, in addition to the high satisfaction of EVs and distributed generation resources, can on the one hand minimize the operating costs and on the other hand reduce and postpone the network strengthening costs.

Funding: No funding for this research.

Data Availability Statement: All data used to support the findings of this study are included within the article.

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

References

[1] S. Shafiee, M. Fotuhi-Firuzabad, and M. Rastegar, "Investigating the Impacts of Plug-in Hybrid Electric Vehicles on Power Distribution Systems," IEEE Transactions on Smart Grid, vol. 4, pp. 1351-1360, 2013.

[2] P.H. Divshali, B.J. Choi, "Electrical market management considering power system constraints in smart distribution grids," Energies, vol. 9, 2016.

[3] Hemmatpour, M. H., Koochi, M. H. R., Dehghanian, P., and Dehghanian, P. (2022). Voltage and energy control in distribution systems in the presence of flexible loads considering coordinated charging of electric vehicles. Energy, 239, 121880.

[4] Bagchi, A. C., Kamineni, A., Zane, R. A., and Carlson, R. (2021). Review and comparative analysis of topologies and control methods in dynamic wireless charging of electric vehicles. IEEE Journal of Emerging and Selected Topics in Power Electronics, 9(4), 4947-4962.

[5] Ghotge, R., van Wijk, A., and Lukszo, Z. (2021). Off-grid solar charging of electric vehicles at long-term parking locations. Energy, 227, 120356.

[6] Scott, M.J, et al., Impacts assessment of plug-in hybrid vehicles on electric utilities and regional US power grids: Part 2: an economic assessment. Pacific Northwest National Laboratory (a), 2007.

[7] B. Falahati, Y. Fu, Z. Darabi, and L. Wu, "Reliability assessment of power systems considering the large-scale PHEV integration," IEEE Vehicle Power and Propulsion Conference, pp. 1-6, 2011.

[8] Su, J., Lie, T. T., and Zamora, R. (2019). Modeling of large-scale electric vehicles charging demand: A New Zealand case study. Electric Power Systems Research, 167, 171-182.

[9] Sheng, M. S., Sreenivasan, A. V., Covic, G. A., Wilson, D., and Sharp, B. (2019, June). Inductive power transfer charging infrastructure for electric vehicles: A new Zealand case study. In 2019 IEEE PELS Workshop on Emerging Technologies: Wireless Power Transfer (WoW) (pp. 53-58). IEEE.

[10] H. Farzin, M. Moeini-Aghtaie, and M. Fotuhi-Firuzabad, "Reliability Studies of Distribution Systems Integrated With Electric Vehicles under Battery-Exchange Mode," IEEE Transactions on Power Delivery, vol. 31, pp. 2473-2482, 2016.

[11] Teng, F., Ding, Z., Hu, Z., and Sarikprueck, P. (2020). Technical review on advanced approaches for electric vehicle charging demand management, Part I: Applications in the electric power market and renewable energy integration. IEEE Transactions on Industry Applications, 56(5), 5684-5694.

[12] Van den Berg, M. A., Lampropoulos, I., and AlSkaif, T. A. (2021). Impact of electric vehicles charging demand on distribution transformers in an office area and determination of flexibility potential. Sustainable Energy, Grids and Networks, 26, 100452.

[13] K. Clement-Nyns, E. Haesen, and J. Driesen, "The impact of charging plug-in hybrid electric vehicles on a residential distribution grid," IEEE Trans. Power Syst., vol. 25, no. 1, pp. 371–380, 2010.

[14] C. Pang, M. Kezunovic and M. Ehsani, "Demand side management by using electric vehicles as distributed energy resources," Proc. of IEEE Electric Vehicle Conf. (IEVC), Greenville, SC, March 2012.

[15] Ali, Abdelfatah, et al. "Optimal allocation of inverter-based WTGS complying with their DSTATCOM functionality and PEV requirements." IEEE Trans. Vehicular Technology, vol. 69, no. 5, pp. 4763-4772, Mar 2020.

[16] Su, Wenzhe, et al. "An MPC-based dual-solver optimization method for DC microgrids with simultaneous consideration of operation cost and power loss." IEEE Trans. Power Systems, vol. 36, no. 2, pp. 936-947, 2020.

[17] Al Zishan, Abdullah, Moosa Moghimi Haji, and Omid Ardakanian. "Reputation-based fair power allocation to plug-in electric vehicles in the smart grid." 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS). IEEE, 2020.

[18] Fachrizal, Reza, et al. "Smart charging of electric vehicles considering photovoltaic power production and electricity consumption: A review." ETransportation vol. 4, pp. 100056, 2020.

[19] Sufyan, Muhammad, et al. "Charge coordination and battery lifecycle analysis of electric vehicles with V2G implementation." Electric Power Systems Research, vol. 184, pp. 106307, 2020.

[20] Singh, Jyotsna, and Rajive Tiwari. "Cost-benefit analysis for v2g implementation of electric vehicles in the distribution system." IEEE Trans. Industry Applications, vol. 56, no. 5, pp. 5963-5973, 2020.

[21] Alsharif, Abdulgader, et al. "A comprehensive review of energy management strategy in Vehicle-to-Grid technology integrated with renewable energy sources." Sustainable Energy Technologies and Assessments, vol. 47, pp. 101439, 2021.

[22] Chang, Shuo, Yugang Niu, and Tinggang Jia. "Coordinate scheduling of electric vehicles in charging stations supported by microgrids." Electric Power Systems Research, vol. 199, pp. 107418, 2021.

[23] Wu, Chuanshen, et al. "A model predictive control approach in microgrid considering multi-uncertainty of electric vehicles." Renewable Energy, vol. 163, pp. 1385-1396, 2021.

[24] Ramadan, Ashraf, et al. "Scenario-based stochastic framework for optimal planning of distribution systems including renewable-based DG units." Sustainability, vol. 13, no. 6, pp. 3566, 2021.

[25] T. Wu, Q. Yang, Z. Bao, and W. Yan, "Coordinated Energy Dispatching in Microgrid With Wind Power Generation and Plug-in Electric Vehicles," IEEE Trans. Smart Grid, vol. 4, pp. 1453-1463, 2013.

[26] Zakariazadeh, Alireza, Shahram Jadid, and Pierluigi Siano. "Integrated operation of electric vehicles and renewable generation in a smart distribution system." Energy Conversion and Management, vol. 89 pp. 99-110, 2015.

[27] M. Moeini-Aghtaie, A. Abbaspour, M. Fotuhi-Firuzabad, and P. Dehghanian, "Optimized Probabilistic PHEVs Demand Management in the Context of Energy Hubs," IEEE Transactions on Power Delivery, vol. 30, pp. 996-1006, 2015.

[28] M. Honarmand, A. Zakariazadeh, and S. Jadid, "Integrated scheduling of renewable generation and electric vehicles parking lot in a smart microgrid," Energy Conversion and Management, vol. 86, pp. 745-755, 2014.

[29] Y. M. Atwa, E. F. El-Saadany, M. M. A. Salama, and R. Seethapathy, "Optimal Renewable Resources Mix for Distribution System Energy Loss Minimization," IEEE Transactions on Power Systems, vol. 25, pp. 360-370, 2010.

[30] A. Yona, T. Senjyu, and T. Funabashi, "Application of Recurrent Neural Network to Short-Term-Ahead Generating Power Forecasting for Photovoltaic System," IEEE Power Engineering Society General Meeting, pp. 1-6, 2007.

[31] G.Boyle, Renewable energy, Oxford, U.K.: Oxford Univ, Press, 2004.

[32] B. S. Borowy and Z. M. Salameh, "Optimum photovoltaic array size for a hybrid wind/PV system," IEEE Transactions on Energy Conversion, vol. 9, pp. 482-488, 1994.

[33] D. Bertsimas and J. N. Tsitsiklis, "Introduction to Linear Optimization," 1997.

[34] R. Billinton and S. Jonnavithula, "A test system for teaching overall power system reliability assessment," IEEE Transactions on Power Systems, vol. 11, pp. 1670- 1676, 1996.

[35] C. Chen, S. Duan, T. Cai, B. Liu, and G. Hu, "Smart energy management system for optimal microgrid economic operation," IET Renewable Power Generation, vol. 5, pp. 258-267, 2011.

[36] S. W. Hadley and A. A. Tsvetkova, "Potential Impacts of Plug-in Hybrid Electric Vehicles on Regional Power Generation," The Electricity Journal, vol. 22, pp. 56-68, 2009.

[37] N. M. M. Razali and A. H. Hashim, "Backward reduction application for minimizing wind power scenarios in stochastic programming," International Power Engineering and Optimization Conference, pp. 430-434, 2010.

List of symptoms:

Binary Variables	
Electric vehicle charging and discharging status v at t hour	$X(v,t)/Y(v,t)$
Generator on or off status j at t hour	$U(j,t)$
Continuous variables	
Electric vehicle discharge power v at t hour	$P_{EV}^{Dcharge}(v,t)$
Electric vehicle charging power v at t hour	$P_{EV}^{Charge}(v,t)$
Electric vehicle energy v at t hour	$E_S(v,t)$
Optimal charge profile for V electric vehicle	$P_{Des}^{Charge}(v)$
Optimal discharge profile for V electric vehicle	$P_{Des}^{Dcharge}(v)$
The number of switching states from charging to discharging or vice versa for a V electric vehicle	D^v
Power purchased from the main grid per t hour	$P_{NTW}(t)$
Wind turbine power per t hour	$P_w(t)$
Power of the photovoltaic system per t hour and s scenario	$P_{pv}(t)$
The output power of distributed generation source j at t hour	$P_{DG}(j,t)$
Active power loss per t hour	$P_{LOSS}(t)$
Radiation intensity per t hour	I_r^r
Network load per t hour	$P_{LOAD}(t)$
Electricity market price per t hour	$Pr_{MRT}(t)$
Price of electric vehicle charging per t hour	$Pr_{EV}^{Charge}(t)$
Price of electric vehicle discharge per t hour	$Pr_{EV}^{Dcharge}(t)$
Price of power loss per t hour	$Pr_{LOSS}(t)$
Active power injected into the bus n per t hour	$P_n(t)$
Reactive power injected into the bus n per t hour	$Q_n(t)$
The power transferring through the transformer n per t hour	$P_{TRANS}(n,t)$
Wind turbine speed per t hour	V_f^t
Objective function related to the profit of electric vehicles	$f_{1,1}$
Objective function related to the profit of distributed production resources	$f_{1,2}$
The objective function of operating costs	f_2
Voltage angle in the bus n per hour t	$\delta_n(t)$
The size of the array (m, n) in the network admittance matrix	$\lvert\gamma_{n,m}\rvert$
The angle of the array (m, n) in the network admittance matrix	$\theta_{n,m}$
Parameters	
Wind turbine rated power	P_{rated}
Wind turbine rated speed	V_r
Low cutoff speed of wind turbine	V_{ci}
High cutoff speed of wind turbine	V_{co}
Efficiency coefficient of a photovoltaic system	η^{PV}
The entire surface of the photovoltaic system	S^{PV}
Ambient temperature	T_a
Efficiency coefficient of electric V vehicle charging	η_v^{Charge}
Efficiency coefficient of electric V vehicle discharge	$\eta_v^{Dcharge}$
Maximum dischargeable power of the V electric vehicle	$P_{Dcharge,v}^{Max}$
Maximum chargeable power of the V electric vehicle	$P_{Charge,v}^{Max}$
The optimal charging level of the V electric vehicle	SOC_{des}^v
The initial charge level of the V electric vehicle's	$SOC_{initial}^v$
Maximum limit of the battery capacity of the electric vehicle	ψ_v^{Max}
Minimum limit of the battery capacity of the electric vehicle	ψ_v^{Min}

Maximum usable percentage of the V electric vehicle's battery capacity	φ_v^{Max}
Minimum usable percentage of the V electric vehicle's battery capacity	φ_v^{Min}
The battery capacity of V electric vehicle	$E_{BatCap,v}$
The weighting factor related to electric vehicle charging	K_{charge}
The weighting factor related to the discharge of electric vehicles	$K_{Dcharge}$
Maximum number of switching states from charge state to discharging or vice versa for any electric vehicle	NS^{Max}
Maximum allowed power that can be received from the upstream network	P_{NTW}^{Max}
Maximum capacity of the transformer	P_{TRANS}^{Max}
The apparent power flowing from the bus n to m per hour t	$S(n,m,t)$
Maximum lines capacity	$S_{n,m}^{Max}$
The voltage on the bus n	$V(n,t)$
Minimum and maximum voltage allowed on the bus n	V_n^{Min}/V_n^{Max}
The maximum output power of the distributed generation source j	$P_{DG,j}^{Max}$
The minimum output power of distributed generation source j	$P_{DG,j}^{Min}$
The cost of setting up a distributed generation resource j	Sc_j
Coefficients of the cost function of the distributed production source j	$a(j),b(j)$
Power increase rate of distributed generation source j	RUP_{DG}^j
The power reduction rate of distributed generation source j	RDN_{DG}^j
Shape factor	α, K
Scale factor	β, c
The length of the time interval	Δt
Collections	
The index corresponding to the source number of distributed generation	j
Index related to the number of network buses	n, m
Index related to optimization time intervals	t
The index corresponding to the scenario number	s
Index related to the number of electric vehicles	v

Journal of Optimization of Soft Computing (JOSC)

Vol. 2, Issue 4, pp: (56-65), Winter-2024
Journal homepage: https://sanad.iau.ir/journal/josc

Paper Type (Research paper)

Optimal use of photovoltaic systems in the distribution network considering the variable load and production profile of Kerman city

Fahimeh Sayadi Shahraki[1], Shaghayegh Bakhtiari Chehelcheshmeh[2*] and Alireza Zamani nouri[3]

1. Department of Electrical Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran.
2. Department of Computer Engineering, Shahrekord Branch, Islamic Azad University, Shahrekord, Iran.
3. Department of Civil Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran.

Article Info

Article History:
Received: 2025/01/01
Revised: 2025/03/15
Accepted: 2025/04/05

DOI:

Keywords:
Photovoltaic systems, Optimization, distribution network, power quality

**Corresponding Author's Email Address:*
sh.bakhtiari@iaushk.ac.ir

Abstract

Photovoltaic systems are very important renewable energy sources, and optimal use of their active and reactive power capacity is very useful in improving the power quality of the distribution network. Therefore, it is necessary to determine the optimal location, number, and capacity of the solar system with appropriate optimization methods so that the maximum reduction in network losses is achieved while considering power quality constraints. Given the complexity and many limitations of the problem, the need to use an appropriate optimization method is evident. In this paper, using the P-PSO optimization algorithm, in the IEEE 33-bus test network, the location and capacity of the active and reactive power of the solar system are determined based on the variable load profile of the network and the daily production curve of the solar system in Kerman city to minimize losses and improve the voltage profile of the electrical energy distribution networks. To increase the accuracy of this optimization, each of the load and production curves is divided into three different levels, according to the geographical climate of Kerman city, in one year, and to evaluate the performance of the proposed method, the relevant results in four different scenarios are examined. The optimization results indicate a significant impact on improving power quality indicators in the presence of photovoltaic systems, especially when using the active and reactive power capacities of these units simultaneously.

1. Introduction

Increasing air pollution and shortage of fuel for fossil power plants in the world have led to an increasing interest in clean and renewable sources. On the other hand, the electrical energy distribution network has faced an increasing load demand, which highlights the need to use local production sources. Solar or photovoltaic systems are among the renewable distributed generation sources that include various advantages such as environmental compatibility, flexibility, reliability, and economic benefits [1]. So far, many studies have been conducted on the connection of distributed generation to the distribution network, and many countries are turning to these sources due to the environmental, economic, and reliability benefits of these systems [2]. Research shows that the way of use, type, capacity, and installation location of these sources are very important in their efficiency in improving the conditions of the distribution system and power quality parameters [3]. Failure to use properly and improper determination of the capacity and installation location of these sources can even lead to a decrease in power quality in the distribution network [4]. In [5-8], the location of distributed generation in the distribution network has been carried out based on various optimization algorithms to reduce losses. The objective function in [5-6] is to reduce losses and costs of distributed generation units. In reference [6], in addition to power losses and installation costs of distributed generation sources, the cost of air pollution is also included in the objective function. In [7-8], power quality constraints have also been evaluated in the process of installing and operating new and renewable energy sources, and the optimal location of distributed generation sources has been carried out by an optimization problem. Although the load and generation profiles of all distributed generation sources in both studies are considered constant in the distribution network. In [9], the effective use of solar systems in a distribution network that is faced with an increase in demand has been investigated. The active power generation capacity has been calculated using environmental conditions, but the reactive power capacity of these sources has not been used. In [10], recommendations and guidelines for the location and capacity of solar system installation in the existing network have been provided for power companies. Given the complexity of the optimization problem in many past studies, this problem has been modeled with various optimization methods, and researchers have tried to optimize their response using newer and more effective optimization methods and considering more constraints [11-13]. In [11], the optimization of the location of distributed generation units is proposed using improved optimization techniques, and finally, the efficiency of the proposed method compared to traditional methods has been shown.

In [12], the optimal placement of distributed generation was carried out by considering high harmonic loads in the network, and the harmonic distortion index was also stated as one of the constraints of the problem. The goal of optimization is to improve the power quality indicators in the system, and only the active power capacity of photovoltaic systems was used. In [14], the capacity and location of solar distributed generation were optimized. To limit the search space, the sensitive buses of the system were initially determined through sensitivity analysis. The load change profile and the production rate of the solar system were not considered in the presented model. In the reviewed studies, the load profile of the network and the production of distributed sources were not considered, but in some studies such as [15], the information on the consumed load in 24 hours and the average active power produced by the solar system were used to determine the optimal location and capacity of the solar system to reduce losses and reduce voltage deviation.

A review of previous studies reveals the following weaknesses:

1- In studies of the use of distributed generation sources from the point of view of the power quality of the distribution network, the use of the average load profile and the average production power of hypothetical distributed generation systems is completely unattainable, and naturally, due to the variability of the load profile and production, the use of the results in practice will not be very effective.

2- In some distributed generation sources, including photovoltaic systems, it is possible to use reactive power capacity if accurately modeled and the available range is determined, which has often been ignored in previous studies.

In the present study, to determine the installation location and the required active and reactive power utilization capacity of photovoltaic sources, the change in the actual annual load profile in Kerman has been considered along with accurate information on the active power production rate of the existing solar system in Kerman. The leveling method takes into account different levels of annual load and production, and as a result of the intersection of these levels, all different load and production level states are extracted, and optimization is carried out based on all levels to reduce losses and network voltage deviation. The answer to the optimization problem is the active and reactive power capacity and the location of the solar system at each load level. Also, the grid voltage constraints, the active and reactive power capacity of the photovoltaic system, and the total power generated based on demand are also considered in solving the optimization problem.

The rest of the paper is organized as follows. In section 2 solution method consist of objective function and constraints formulations are presented. Section 3 Describes how to implement the proposed method of intersecting load and production levels and P-PSO method. Simulation scenarios and results are provided in section IV and section V discusses the results and concludes the paper.

2. Solution method

2.1. Objective function

The goal of optimization is to reduce active losses and maintain the bus voltage profile within the desired range. The objective function F is defined as equation (1) which must be minimized. The weighting coefficients k_1 and k_2 are chosen between 0 and 1 and their sum is equal to one and shows the degree of influence of each objective function on the overall objective function.

$$F = k_1 f_1 + k_2 f_2 \tag{1}$$

The first objective function f_1 is the total real losses in the buses, which is obtained from equation (2). The second objective function f_2 is the improvement rate of the voltage profile, which is defined as the sum of the squares of the bus voltage difference to the nominal value of one per unit, and is obtained from equation (3).

$$f_1 = \sum_{i=1}^{N_{bus}} \frac{Ploss_i \ with\ out\ DG}{Ploss_i \ with\ DG} \tag{2}$$

$$f_2 = \sum_{i=1}^{N_{bus}} (v_i - v_{nom})^2 \tag{3}$$

$Ploss$ is the total bus losses and N_{bus} is the number of busses.

In order to normalize the objective function of the problem, in equation (2), the total bus losses without installing distributed generation sources are divided by the total losses in the presence of distributed generation sources. $v_i$is the voltage of the ith bus, v_{nom} is the nominal voltage in terms of per unit.

2.2. Constraints

2.2.1. Power balance constraint

According to equation 4, where P_{Di}, P_{PVi} are the active power generated by DG and the active power consumed in the nth bus, respectively, and P_L represents the active losses in the network in question. P_{slack} is the transmitted power from the upstream network.

$$\sum_{i=1}^{N} P_{PVi} + P_{slack} = \sum_{i=1}^{N} P_{Di} + P_L \tag{4}$$

2.2.2. Active and reactive power generation range by solar DG

The active generation power limit of the jth distributed generation unit is obtained from equation (5).

$$P_{jPVmin} < P_j < P_{jPVmax} \quad j = 1.2.\dots.N_{DG} \tag{5}$$

N_{DG} is the number of distributed generation, P_{PVmin} is the minimum active power generated, P_{PVmax} is the maximum active power generated. In this paper, the following method is used to obtain the reactive power range:

At any time of the day, the reactive power generated by the photovoltaic system is limited by various constraints depending on the operating point of the system, and the reactive power exchanged between the grid and the photovoltaic system converter. We assume that the PV system with the maximum active power is in the system. One of the most important constraints for the reactive power of the system is determined by the maximum apparent power of the inverter. The reactive power of the PV depends on the maximum voltage and current values of its converter, so to calculate the controllable limit Q_{PV}-P_{PV}, the maximum voltage and current values of the converter $V_{C\prime max}$and $I_{C\prime max}$ must be considered. The relationship between active and reactive power considering the converter current is as follows:

$$P_{PV}^2 + Q_{PV}^2 = (I_C V_{PV})^2 \tag{6}$$

And the relationship between active and reactive power, taking into account the voltage limitation of the converter, is as follows:

$$P_{PV}^2 + (Q_{PV} + \frac{V_{PV}^2}{X_C})^2 = (\frac{V_C V_{PV}}{X_C})^2 \tag{7}$$

Using this relationship, the design value $V_{C\prime max}$ can be calculated, which determines the maximum inverter dc link voltage $V_{dcmax}$and $I_{C\prime max}$ [17-18]. The maximum converter current should be in the following relationship using the values of PV voltage, active and reactive power:

$$I_{C\prime max} = \frac{P_{PV\prime R}^2 + (V_{PV\prime R}\ tan\theta_R}{V_{PV\prime min}} \tag{8}$$

The maximum converter voltage is also calculated from the active and reactive power values and the maximum PV voltage as follows:

$$V_{C\prime max} = \frac{X_C}{V_{PV\prime max}} \sqrt{P_{PV\prime R}^2 + (V_{PV\prime}\ tan\theta_R + \frac{{V_{PV\prime max}}^2}{X_C})^2} \tag{9}$$

Also, the PV reactive power, taking into account the rated current and rated voltage of the PV system, is:

$$Q_{c\prime PV}^t = \sqrt{(V_{PV} I_{C\prime max})^2 - {P_{PV}^t}^2} \tag{10}$$

$$Q_{v\prime V}^t = \sqrt{(\frac{V_{C\prime max} V_{PV}}{X_C})^2 - {P_{PV}^t}^2 - \frac{V_{PV}^2}{X_C}}$$

Finally, at each operating point, the maximum reactive power at each hour t is obtained using the following equation:

$$Q_{PV\prime max}^t = min\{Q_{c\prime PV}^t{}' Q_{v\prime PV}^t\} \tag{11}$$

In this article, $V_{PV\prime max}$=1.05, $V_{PV\prime min}$=0.95, and X_C=0.3 are considered.

3.2.3. Bus voltage limit

In equation (12), the minimum and maximum allowable voltage limits for all buses are considered to be 0.9 to 1.01 per unit. Where V_i is the voltage of the i-th bus, $V_{i.min}$, $V_{i.max}$ are the minimum and the maximum bus voltage respectively.

$$V_{i.min} < V_i < V_{i.max} \quad i = 1.2. \dots . N_{bus} \qquad (12)$$

3. Proposed intersection of load and production levels method and solving by P-PSO

To bring the results of the optimization problem closer to reality, changes in the load curve and solar system production should be included in the problem. In this paper, for more realistic results and to increase the optimization accuracy, changes in the load and production curve are considered in a one-year period. Since temperature changes in the seasons are the most important parameter affecting the load in the short term, the annual load curve of the Kerman city network is divided into three conditions: maximum, minimum, and intermediate temperatures, and as a result, load consumption. Kerman's peak load occurs in the summer season and the high load level occurs in this time period. In winter, due to the city's temperate geographical location and the major use of gas in heating devices, the amount of electricity used is low and the low load level occurs in this period. Finally, in the temperate seasons of autumn and spring, load consumption is considered as the medium load level. Accordingly, the load curve of Kerman city is divided into three low load, high load, and medium load levels in a year based on seasonal changes, as shown in Table 1. Also in Table 2, the active and reactive power capacity of the photovoltaic system is specified for each load level.

Table 1. Network load levels studied in one year

Load Levels	month of the year
Low load	November, December, January, February
Mid load	April, May, October, March
High load	June, July, August, September

Table 2. Active and reactive power of the photovoltaic system for each load level

	Active power	*Reactive power*
Test network	*P*	*Q*
Low load	*0.3P*	*1.5Q*
Mid load	*0.6P*	*0.5Q*
High load	*P*	*Q*

Table 3. Yearly leveling the solar system production curve

Load Level	months	Capacity factor percentage	Probability of occurrence per year
Low load	August, November, December, January	74%	$\frac{4}{12}$
Mid load	April, May, February, March	90%	$\frac{4}{12}$
High load	June, July, September, October	100%	$\frac{4}{12}$

Solar cells rarely operate at their maximum power point because the output power is affected by radiation and ambient temperature. Load changes also affect the shift of the operating point and the power received from the system. By studying the energy output of the solar system in the geographical area of Kerman, the production curve of the photovoltaic system is obtained at three different levels low load, medium load, and high load according to Table 3.

3.1. Intersection of the load and production curves

In this paper, to increase the accuracy of the obtained results, it is used to consider different states of the curve resulting from the intersection of two load curves and the solar system production curve. Since three levels (low load, medium load, and high load) are considered for the load and production curve, the resulting curve has 9 states, but with the assumptions of the problem for the city of Kerman, only 6 of these 9 states occur. The intersection of the two curves is shown in Figure 1.

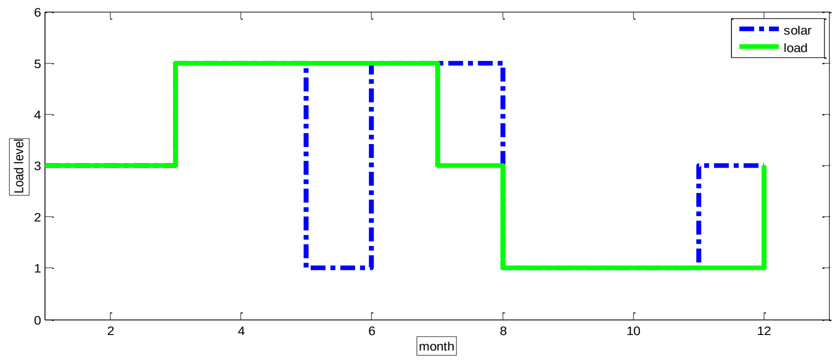


Figure 1. The intersection of two annual load and production graphs of a solar system

The characteristics of the different states of the graph resulting from the intersection of the two load and production curves are presented in Table 4. Accordingly, in order to solve the problem in question, by determining the loss objective function for each of these 6 states, according to equation (13), the objective function of the problem is obtained by considering the changes in the load curve and the changes in the production curve.

$$F_T = \frac{3}{12}g1 + \frac{3}{12}g2 + \frac{3}{12}g3 + \frac{1}{12}g4 + \frac{1}{12}g5 + \frac{1}{12}g6 \qquad (13)$$

First case: For the months of November-December-January, the load curve and the generation curve are at low load level. Therefore, the coefficient of the corresponding loss function $g1$ is obtained as 3/12. The generation capacity factor is 74%. (For load distribution) its active power is considered to be 0.3 times the active power of the test network load and its reactive power is considered to be 1.5 times the reactive power of the test network.

Second case: In the months of June, July, and September, the load curve and the generation curve are at medium load level, therefore the coefficient of the corresponding loss function $g2$ is obtained as 3/12. The generation capacity factor is 90%. (For load distribution) its active power is considered to be 0.6 times the active power of the test network load and its reactive power is considered to be 0.5 times the reactive power of the test network. The table below describes the status of all 6 possible cases.

Table 4. All possible load and production states and 6 possible states

Mid	Mid	High	High	Low	Low	Mid	High	Low	Load Level
High	Low	Mid	Low	Mid	High	Mid	High	Low	Generation Level
$\frac{1}{12}$	0	0	$\frac{1}{12}$	$\frac{1}{12}$	0	$\frac{3}{12}$	$\frac{3}{12}$	$\frac{3}{12}$	Probability
October	Don't occure	Don't occure	August	February	Don't occure	April May March	June July September	November December January	Month
0.6P	0.6P	P	P	0.3P	0.3P	00.6P	P	0.3P	Active power
0.5Q	0.5Q	Q	Q	1.5Q	1.5Q	0.5Q	Q	1.5Q	Reactive power
100%	74%	90%	74%	90%	100%	90%	100%	74%	Capacity factor

3.2. Problem-solving with the P-PSO algorithm

To solve the optimization problem described in this section, a new and very effective and useful algorithm, P-PSO, has been used, the capabilities and implementation of which are shown in [17]. Despite the competitive performance of PSO, it is noted the tendency of PSO swarm to converge prematurely in the local optima, due to its rapid convergence on the best position found so far at the early stage of optimization. Main challenging issue that needs to be addressed is the proper control on the exploration and exploitation searching of PSO.

Basic PSO Algorithm

In basic PSO, each particle that is roaming through the *D* dimensional problem hyperspace represents the potential solution for a specific problem. For particle *i* two vectors, i.e. position vector $\mathrm{X_i} = [\mathrm{X_{i1}}‘\, \mathrm{X_{i2}}‘\, \dots ‘\mathrm{X_{iD}}]$ and velocity vector $\mathrm{V_i} = [\mathrm{V_{i1}}\, ‘\mathrm{V_{i2}}‘\, \dots ‘\, \mathrm{V_{iD}}]$ are used to represent its current state. Additionally, each particle *i* can memorize its personal best experience ever encountered (i.e. cognitive experience), represented by the personal best position vector $\mathrm{P_i} = [\mathrm{P_{i1}}‘\, \mathrm{P_{i2}}\, \dots ‘\, \mathrm{P_{iD}}]$. The position attained by the best particle in the society (i.e. social experience) is represented as$\mathrm{P_g} = [\mathrm{P_{g1}}‘\, \mathrm{P_{g2}}‘\, \dots ‘\, \mathrm{P_{gD}}]$. Mathematically, at iteration (*t* + 1) of the searching process, the *d*-th dimension of particle *i*'s velocity, $\mathrm{V}_{\mathrm{id}}(\mathrm{t}+1)$ and position $\mathrm{X}_{\mathrm{i,d}}(\mathrm{t}+1)$ are updated as follows:

$$V_{id}(t+1) = V_{id}(t) + c_1 r_1\big(P_{id}(t) - X_{id}(t)\big) + c_2 r_2\big(P_{gd}(t) - X_{id}(t)\big) \quad (14)$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \quad (15)$$

Where $c_1$and c_2 are the acceleration coefficients; r_1 and r_2 are two random numbers generated from a uniform distribution within the range of [0, 1]. Particles velocity is clamped to a maximum magnitude of V_{max} to prevent swarm explosion. When minimizing the fitness function f in D dimensional search space, particle *i*'s P_i position in iteration (*t* + 1) is updated as follows [19]:

$$P_i(t+1) = \begin{cases} X_i(t+1) & if\ f\big(X_i(t+1)\big) < f(\,P_i(t)) \\ P_i(t) & other\ wise \end{cases} \quad (16)$$

P-PSO Algorithm

Despite the competitive performance of PSO, researchers have noted the tendency of PSO swarm to converge prematurely in the local optima, due to its rapid convergence on the best position found so far at the early stage of optimization [20]. Once the swarm congregates at such position, little opportunity is afforded for the population to explore for other solution possibilities by designing perturbation module. This leads to the entrapment of the swarm within the local optima of search space and thus premature convergence occurs. Another challenging issue that needs to be addressed is the proper control on the exploration and exploitation searching of the PSO. So P- PSO which was proposed in [17] is characterized by:

3.2.1 Velocity calculation

In this model to achieve better control on the algorithm’s exploration and exploitation capabilities, particles velocity is dependent on both particle’s fitness and time. More specifically, particles with better (i.e. lower) fitness value are assigned with lower ω_i that favour the exploitation, whilst particles with worse (i.e. higher) fitness value is encouraged for the exploration by assigning them with higher ω_i. Mathematically, particle *i*’s inertia weight, i.e. ω_i is calculated as follows:

$$\omega_i = c_1\big((\omega_{max} - \omega_{min}) * G_i + \omega_{min}\big) + c_2\left((\omega_{max} - \omega_{min}) * \frac{maxiter-iter}{maxiter} + \omega_{min}\right) \quad (17)$$

Where ω_{max} and ω_{min} represent the maximum and minimum inertia weights, respectively, i.e. $\omega_{max} = 0.9$ and $\omega_{min} = 0.4$; G_i represents the fitness dependent weight value that determines ω_i of particle *i* as shown:

$$G_i = \frac{f(P_i) - f_{min}}{f_{max} - f_{min}} \quad (18)$$

Where f_{max} and f_{min} represent the maximum and minimum personal best fitness values that exist in the population. Equation (18) shows that the particle with smaller fitness has smaller and thus is assigned with smaller ω_i and vice versa. To this end, we update the particle i's velocity, V_i as follows:

$$V_i(t+1) = \omega_i V_i + \textstyle\sum_{P_k \in N_i} c_k r_k (P_k - X_i) \quad (19)$$

where P_k represents the personal best position of neighboring particles that exist in particle i's neighborhood; N_i represents the number of neighbouring particles available for particle (18) i; c_k represents the acceleration coefficient that equally distributed among the N_i neighboring particles, calculated as, $c_k = c/N_i$ where; $c = 4.1$, r_k represents the random number in the range of [0, 1].

3.2.2 Perturbation module

To alleviate the premature convergence issue, a perturbation module is adopted to perform perturbation on the P_g particle and provide extra diversity for it to jump out from local optima, if its fitness is not improved for m successive function evaluations (FEs). The m value that used to trigger perturbation module should not be set too large or too small, as the former wastes the computation resources, whilst the latter degrades algorithm’s convergence speed. Herein, m is set as 5. In perturbation module, one of the d-dimension of P_g particle i.e. P_{gd} is first randomly selected and it is then perturbed randomly by a normal distribution as follows:

$$P_{gd}^{per} = \begin{cases} P_{gd} + r_4\big(X_{max,d} - X_{min,d}\big) & if\ r_3 > 0.5 \\ P_{gd} - r_4\big(X_{max,d} - X_{min,d}\big) & if\ r_3 \le 0.5 \end{cases} \quad (20)$$

Where P_{gd}, is the perturbed P_g; r_3 is a random number with the range of [0, 1] and generated from uniform distribution; r_4 is a random number generated from the normal distribution of $N\sim(\mu,\sigma^2)$ with mean value of $\mu = 0$ and standard deviation of $\sigma = R$, respectively. R represents the perturbation range that linearly decreased with the number of FEs as follows:

$$R = R_{max} - (R_{max} - R_{min})\frac{fes}{FE_{max}} \qquad (21)$$

Where R_{max} and R_{min} are the maximum and minimum perturbation ranges, respectively; fes is the FEs number used; max FE is the predefined maximum FEs. The newly perturbed P_g particle, i.e. P_g^{per} is then evaluated and examined. It will replace P_g if $f(P_g^{per}) < f(P_g)$.

The process of solving the aforementioned optimization problem is shown in the flowchart in Figure 2.

4. Numerical results

The maximum number of distributed generation sources is four and the generation capacity range of each DG unit is 1.2 MW and 1.2 MVAR.

To show the effect of using the reactive power capacity of PV systems and also to evaluate the proposed method on the selection of optimal location and capacity, simulations(18)have been performed considering four scenarios:

1- The studied network without the presence of distributed generation such as photovoltaic systems

2- The network in the presence of photovoltaic systems and assuming only the use of active power

3- The network in the presence of photovoltaic systems and assuming the production of active and reactive power

4- Similar to case 3 but without leveling (assuming the average load and production).

In this paper, the role of reducing losses and improving the voltage profile in the equal objective function is considered (K١=K2=0.5).

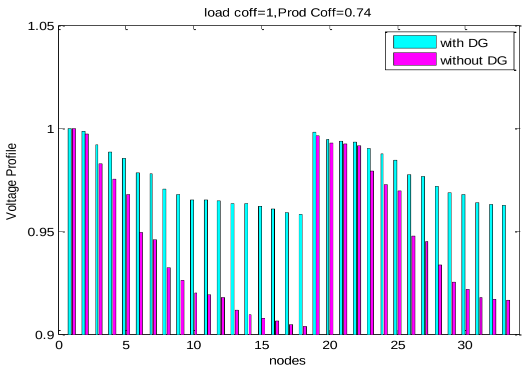


Figure 1. Intersection of two annual

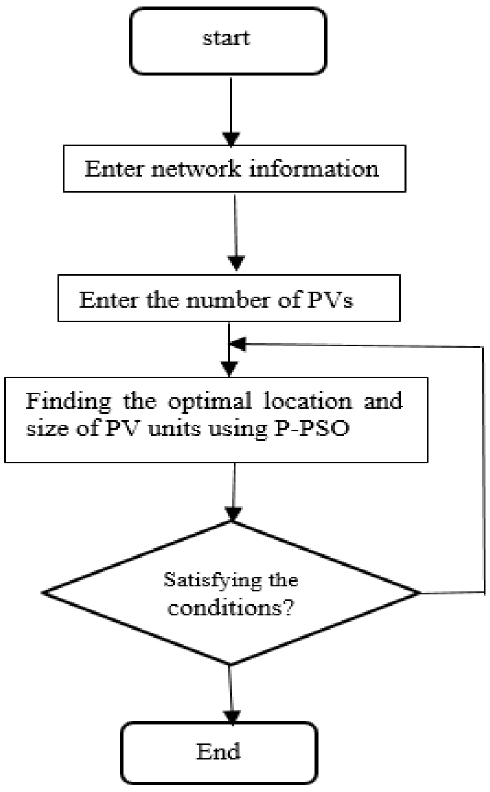

start
Enter network information
Enter the number of PVs
Finding the optimal location and size of PV units using P-PSO
Satisfying the conditions?
End

Figure 2. Problem-solving algorithm

Table 5. Voltage deviation changes in four scenarios

Average voltage deviation		**6**	**5**	**4**	**3**	**2**	**1**	**states**
	Load factor	0.3	0.6	1	0.3	0.6	1	**Scenario**
	Generation factor	0.9	1	0.74	0.74	0.9	1	

	V.D6	V.D5	V.D4	V.D3	V.D2	V.D1	
	-	0.0599	0.0402	0.1338	0.0599	0.0402	**1**
0.1947	0.1579	0.0314	0.4052	0.2500	0.0652	0.2654	**2**
0.0541	0.0002	0.0036	0.0245	0.0021	0.0012	0.0077	**3**
0.0583	0.0105	0.18	0.1374	0.0114	0.0847	0.028	**4**

Table 6. Capacity and location of installed PV units

PV 4		PV 3		PV 2		PV 1		Scenario
Capacity	Location	Capacity	Location	Capacity	Location	Capacity	Location	
P(kW) Q(kVAR)		P(kW) Q(kVAR)		P(kW) Q(kVAR)		P(kW) QkVAR		
-	-	-	-	-	-	-	-	1
473.40	31	444.43	14	642.82	6	661.04	24	2
0		0		0		0		
631.72	24	439.78	7	451.3	14	576.7	30	3
388.59		391.38		288.56		759.21		
741.66		588.7		535.6		953.4		
448.11	14	516.43	30	649.20	6	670.63	24	4
329.59		966.1		535.63		480.81		

Table 7. Power loss in four scenarios

$PLOSS_{MID}$-	$PLOSS_{ave}$	Load 0.3 Generation 0.9	0.6 1	1 0.74	0.3 0.74	0.6 0.9	1 1	Scenario
		F6	F5	F4	F3	F2	F1	
	-	159.52	63.91	210.99	159.52	63.91	210.99	1
	0.5355	0.9575	0.286	0.475	0.9186	0.2688	0.3816	2
	0.1888	0.2655	0.1987	0.2099	0.3074	0.1289	0.0941	3
0.1124	0.2136	0.1899	0.4232	0.1709	0.2188	0.2883	0.0858	4

The results of Table 5 show that the optimal use of active and reactive power capacity can be effective in reducing voltage deviation. So scenario 3 shows the lowest voltage deviation. Table 6 shows the results of the capacity and location of installed PV units. To evaluate the results from the point of view of losses, Table 7 summarizes the results of the different objective functions and also reports the average losses. As can be seen in scenario 1, losses occur at the highest level and the use of the active power capacity of the photovoltaic system leads to a significant reduction in the level of power losses in the network. Also, comparing the results of the second and third scenarios shows the effect of using the reactive power capacity of the photovoltaic system in reducing losses. So at peak load, losses are reduced to about one-third. To see the effect of using the photovoltaic system, the voltage results of all buses at peak load in the first and third scenarios are shown in Figure 3. According to the figure, in the third scenario, due to the simultaneous use of the active and reactive power capacity of the solar system, the bus voltage deviation level is minimized. The results of Table

6 show that the use of reactive and active power capacity simultaneously with the proposed leveling method is effective in controlling the bus voltage deviation, reducing losses, and reducing system costs. To see the importance of using the proposed leveling method, the results of the third and fourth scenarios can be compared. As can be seen, the voltage deviation at peak load in the third scenario is about one-sixth of that in the fourth scenario, and the power losses in the third scenario are less than half of those in the fourth scenario.

These results reveal the importance of load and generation leveling according to climate in the use of distributed generation resources.

Figure 4 shows the convergence plot of the P-PSO method compared to PSO. Comparing the two plots reveals the optimization quality and escape from the local convergence of P-PSO.

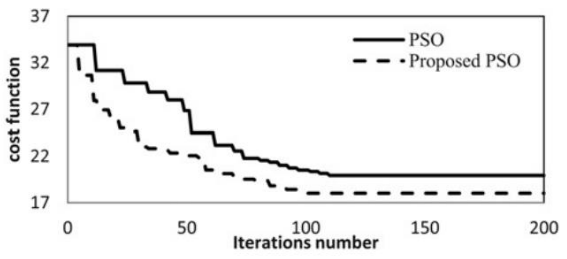


Figure 4. Convergence diagram of P-PSO and PSO methods

5. Conclusion

In this paper, the location and capacity of the active and reactive power of the photovoltaic system in the distribution network were optimized based on the load leveling required and the generation capacity of the photovoltaic system. The aim of optimizing the voltage and reactive power control in the network under study was to reduce active power losses and bus voltage deviations as the main objectives. To use the leveling method, the actual load and generation profile of Kerman was used, and the results obtained indicate the importance of proper use of distributed generation resources and the advantage of using the reactive power capacity of these systems. Observation of the results shows that the use of the photovoltaic system leads to a profound reduction in active losses and bus voltage deviations in the system. However, using the reactive power capacity of these resources compensates the system voltage level more appropriately. Of course, it is necessary to consider the limitations of the active and reactive power generated by these distributed generation resources. Also, the use of the P-PSO optimization method shows the appropriate quality of optimization of this method and the escape from local convergence in complex problems. The results of the article show that the appropriate use of the leveling method in a distribution system with distributed generation resources will lead to improved results and achieve the goal of improving power quality in the network.

References

[1] M. C. V. Suresh and E. J. Belwin, “Optimal DG placement for benefit maximization in distribution networks by using dragonfly algorithm,” Renewables Wind Water and Solar, vol. 5, pp. 2–8, 2018.

[2] T. D. Pham, T. T. Nguyen, and B. H. Dinh, “Find optimal capacity and location of distributed generation units in radial distribution networks by using enhanced coyote optimization algorithm,” Neural Computing & Applications, vol. 33, pp. 4343–4371, 2021.

[3] A. Rathore and N. P. Patidar, “Optimal sizing and allocation of renewable based distribution generation with gravity energy storage considering stochastic nature using particle swarm optimization in radial distribution network,” Journal of Energy Storage, vol. 35, p. 102282, 2021.

[4] P. C. Chen, V. Malbasa, Y. Dong, and M. Kezunovic, “Sensitivity analysis of voltage sag-based fault location with distributed generation,” IEEE Transactions on Smart Grid, vol. 6, no. 4, pp. 2098–2106, 2015.

[5] M. Kashyap, A. Mittal, and S. Kansal, “Optimal placement of distributed generation using genetic algorithm approach,” Lecture Notes in Electrical Engineering, vol. 476, pp. 587–597, 2019.

[6] S. R. Ramavat, S. P. Jaiswal, N. Goel, and V. Shrivastava, “Optimal location and sizing of DG in the distribution system and its cost-benefit analysis,” Advances in Intelligent Systems and Computing, vol. 698, pp. 103–112, 2019.

[7] T. S Tawfeek, A. H. Ahmed, and S. Hasan, “Analytical and particle swarm optimization algorithms for optimal allocation of four different distributed generation types in radial distribution networks,” Energy Procedia, vol. 153, pp. 86–94, 2018.

[8] S. Mirsaeidi, S. Li, S. Devkota et al., “Reinforcement of power system performance through optimal allotment of distributed generators using metaheuristic optimization algorithms,” Journal of Electrical Engineering & Technology, vol. 17, no. 5, pp. 2617–2630, 2022.

[9] Adeagbo, A.; Olaniyi, E.; Ofoegbu, E.; Abolarin, A. Solar photo-voltaic system efficiency improvement using unitary-axis active tracking system. Int. J. Sci. Eng. Res. 2020, 11, 502–508.

[10] Adewuyi, O.B.; Ahmadi, M.; Olaniyi, I.O.; Senjyu, T.; Olowu, T.O.; Mandal, P. Voltage security-constrained optimal generation rescheduling for available transfer capacity enhancement in deregulated electricity markets. Energies 2019, 12, 4371.

[11] Hemeida MG, Ibrahim AA, Mohamed AA, et al. Optimal allocation of distributed generators DG

based Manta Ray Foraging Optimization algorithm (MRFO). Ain Shams Eng J, 2021, 12:609–619.
[12] Shradha SinghPariharNitinMalik, Analysing the impact of optimally allocated solar PV-based DG in harmonics polluted distribution network, Sustainable Energy Technologies and Assessments Volume 49, February 2022.
[13] Abou El-Ela AA, El-Sehiemy RA, Abbas AS. Optimal placement and sizing of distributed generation and capacitor banks in distribution systems using water cycle algorithm. IEEE Syst J, 2018, 12:3629-3636.
[14] Varaprasad Janamala, K Radha Rani, Optimal allocation of solar photovoltaic distributed generation in electrical distribution networks using Archimedes optimization algorithm Clean Energy, Volume 6, Issue 2, April 2022, Pages 271–287.
[15] G. Guerra, J. a Martinez, and S. Member, "A Monte Carlo Method for Optimum Placement of Photovoltaic Generation Using a Multicore Computing Environment," PES Gen. Meet. Conf. Expo. 2014 IEEE. IEEE, pp. 1–5, 2014.
[16] Fahimeh Sayadi, Saeid Esmaeili, Farshid Keynia, Two-layer volt/var/total harmonic distortion control in distribution network based on PVs output and load forecast errors, IET Generation, Transmission & Distribution, Volume 11,2016.
[17] Fahimeh Sayadi, Saeid Esmaeili, Farshid Keynia, Feeder reconfiguration and capacitor allocation in the presence of non-linear loads using new P-PSO algorithm, IET Generation, Transmission & Distribution, Volume 10, 2015.
[18] Calderaro V, Conio G, Galdi V, Massa G, Piccolo A. Optimal decentralized voltage control for distribution systems with inverter-based distributed generators. IEEE Trans Power Syst 2014; 29:230–41.

[19] Van den Bergh, F., Engelbrecht, A., A Cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation: 225-239, 2004.