Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 279–287
DOI: 10.15439/2024F8403
ISSN 2300-5963 ACSIS, Vol. 39

Virtual Power Plant Optimization Service - Benchmark of Solvers

Filipe Alves[0000−0002−8387−391X*], Rui Ribeiro[0009−0002−3421−8932*], Maria Petiz[0009−0001−1059−0871*], Ali Abbasi[0000−0002−5581−1279*], Pedro Carvalho[0009−0002−1319−6535*], Ricardo Faia[0000−0002−1053−7720†], Pedro Faria[0000−0002−5982−8342†], Zita Vale[0000−0002−4560−9544†], and Ricardo Rodrigues[0000−0001−7986−3754*]

* DTx — Digital Transformation CoLAB, University of Minho, 4800-058 Guimarães, Portugal
Email: {filipe.alves, rui.ribeiro, maria.petiz, ali.abbasi, pedro.carvalho, ricardo.rodrigues}@dtx-colab.pt

† GECAD - Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, LASI - Intelligent Systems Associate Laboratory; Polytechnic of Porto; Porto, Portugal
Email: {rff, pnf, zav}@isep.ipp.pt

***Abstract*—This work provides a comprehensive analysis of the optimization of a Virtual Power Plant (VPP), that consider the presence of energy storage systems and controllable loads, through the benchmarking of various solvers. It delves into the development of a Mixed Integer Linear Programming (MILP) algorithm aiming at optimizing energy management and exchange within a VPP, that takes into account the operation of shift electric appliances and battery storage systems among different houses. The proposed model aims to minimize the overall electricity cost while ensuring that the energy demand of the system is met, the battery state of charge is maintained within safe operating limits, and the shift electrical appliance is scheduled. Furthermore, the experimental comparisons, the study evaluates the performance of commercial and open-source solvers in handling the complex dynamics of energy demand and supply. The findings highlight the importance of solver selection in enhancing the management, scalability, and reliability of VPP optimization strategies, offering insights into the optimal combination of programming interfaces and solvers for efficient VPP operation.**

I. Introduction

AROUND the world, Renewable Energy Sources (RESs) have taken advantage of the strong development of Distributed Energy Resources (DERs) [1]. A solution to this challenge is to aggregate the RESs, assuming that there may be unstable output and inconsistent generation of individual RES to emerge like a conventional generator with relatively stable output. Virtual Power Plants (VPPs) provide the potential solution for this problem, integrating Cyber-Physical Systems (CPS) to enhance the efficiency and coordination of various distributed energy resources [2].

The concept of VPPs represent a transformative approach in the energy sector, aiming to integrate various DERs such as renewable energy sources, controllable loads, and Energy Storage Systems (ESSs) into a cohesive and optimized network. This integration is facilitated through advanced software and hardware technologies, allowing for centralized control while maintaining the autonomy of individual resources. The concept of VPPs is becoming increasingly important as the global shift towards RESs intensifies, requiring innovative solutions to manage the variability and decentralization inherent in these sources [3].

In this sense, based on strategies and technologies for monitoring, controlling and programming DERs, it is possible for a VPP to generate benefits such as decreasing the customer's energy cost, reducing emissions, increasing energy efficiency, and asset control/optimization [4]. In this paper, a benchmark study for a VPP algorithm is addressed, where energy balance is guaranteed, the battery's state of charge is kept within safe operating limits, and electrical appliances are shifted accordingly to the user's requirements, all with a view to minimize operating costs. The scenario that served as the basis for this work was based on the model formulated in [5]. This deals with energy consumption management in the residential sector, as it is crucial to mitigate peak demand. It is based on reprogramming household appliances to change their load during peak hours, which significantly helps the grid. By combining the capabilities of ESS and load-shifting appliances, the Home Energy Management System (HEMS) can intelligently program and coordinate the operation of these devices to maximize the use of renewable energy resources and minimize electricity costs.

Moreover, this paper explores the development of a MILP algorithm designed to optimize the energy exchange among different houses within a VPP, focusing on the decision-making process regarding energy trading in specific time frames. The need for such optimization arises from the complex dynamics of energy demand and supply, the integration of RESs, and the economic considerations of buying and selling energy in the competitive market. The study delves into the performance of various commercial and open-source solvers in handling the optimization model. This comparative analysis is crucial for identifying the most effective computational tools for VPP optimization, taking into account factors such as computational efficiency, scalability, and the ability to handle VPP operations.

This paper is organized as follows: Section II presents the

This article is a result of the Innovation Pact "NGS - New Generation Storage" (reference 58), co-financed by NextGeneration EU, through the Incentive System "Agendas para a Inovação Empresarial" ("Agendas for Business Innovation"), within the Recovery and Resilience Plan (PRR). The GECAD team authors acknowledge the work facilities and equipment provided by GECAD research center (UIDB/00760/2020) to the project team.

literature review on VPPs, highlighting their significance, operational challenges, and the role of optimization algorithms. Section III describes the optimization algorithm used in VPP, detailing the mathematical model, objectives, parameters, and key constraints. Section IV discusses the practical implementation of the optimization service, including the architecture, data workflow, and comparison between commercial and open-source solvers. Section V presents the results of the study, focusing on the input/output data, test scenarios, implementation, and discussion. Finally, Section VI concludes the paper and suggests directions for future work.

II. Literature Review

The emergence of VPPs marks a significant evolution in the power system architecture, aggregating DERs such as distributed generation, energy storage, and flexible charging capabilities to provide vital grid services [6], [7]. The integration of VPPs into the energy system enhances efficiency, reliability, and sustainability, addressing the challenges posed by the increasing penetration of variable RESs like solar and wind energy. Since the early 2000s, the concept of VPPs has gained prominence as a solution to the variability and unpredictability of RESs, which, despite their clean and renewable nature, introduce issues related to system stability and reliability [2].

To further enrich our understanding of VPPs and their pivotal role in integrating DERs, it's crucial to explore the operational challenges they face in greater depth, particularly in terms of reliability, scalability, and security. VPPs, being at the forefront of the transition to a decentralized energy system, face unique challenges related to the intermittent nature of RESs, the complexity of managing diverse energy assets, and ensuring cybersecurity in an increasingly digitalized infrastructure. VPPs offer a strategic response to these challenges by enhancing system flexibility. They play a crucial role in balancing the generation from RESs with demand, providing essential grid services such as frequency re-balancing, load management, and participating in energy markets. This ability to mitigate the impacts of variable energy sources on the grid underlines the importance of developing advanced optimization algorithms for VPP operation [4].

When exploring the development and optimization of VPPs, the importance of Cyber-Physical Systems (CPS) in smart grids becomes evident [8]. The integration of such systems is crucial for the efficiency of VPPs, as it facilitates communication and coordination between the various DERs, from generation to consumption. This context lays the foundation for understanding VPPs not just as technological entities, but as complex energy ecosystems that require advanced optimization to operate effectively.

The optimization of VPPs, especially with regard to the control of microgrids, is an area of growing interest in the scientific community [9]. Effective control of microgrids, essential components of VPPs, plays a significant role in the management of DERs, highlighting the need to develop robust algorithms that can deal with the complexity and dynamism of the energy system. The approach of using MILP to optimize DERs in microgrids illustrates the applicability of this methodology in solving complex optimization problems in VPPs [10]. The ability to model sizing, allocation and operation decisions in an integrated way offers a powerful tool to optimize the performance of VPPs, ensuring energy efficiency and sustainability. The objective is to minimize energy usage costs in a day-ahead operation. Furthermore, on the application of MILP in integrated power systems provides valuable insights into the different interfaces and solvers used in optimizing VPPs ([11]). This analysis highlights the diversity of available tools and the importance of selecting the most appropriate approach for each specific scenario, emphasizing the need for detailed benchmarks that evaluate the performance of these different solvers.

More recently, some works provide insights into the current trends and methodologies in VPP optimization, including the application of MILP models, consideration of trade markets, and the optimization of solvers to enhance VPP operations [12], [13]. Recent studies have underscored the importance of addressing these challenges through advanced optimization strategies, real-time data analytics, and robust cybersecurity protocols. For instance, the integration of machine learning algorithms for predictive analysis can significantly enhance the forecasting accuracy of renewable energy production and consumption patterns, thus improving the VPP's ability to balance supply and demand effectively [14], [15].

To ensure the resilience of VPP operations, research has also focused on developing sophisticated strategies that can dynamically adapt to changes in the energy market and regulatory environments. This includes the application of adaptive optimization algorithms that can accommodate multiple objectives, such as minimizing costs, maximizing the use of renewable energy, and ensuring grid stability. The contribution of this article lies in the application of a MILP algorithm developed specifically to optimize energy exchanges within VPP, addressing the complexity inherent in energy purchase and sale decisions taking into account the shifts of home appliances' energy management. By comparing the performance of various solvers and interfaces, this research not only fills an identified gap in the existing literature, but also offers practical guidelines for effectively implementing optimization solutions in VPPs. This approach allows for a deeper understanding of the capabilities and limitations of available solvers, guiding future research and development in optimizing VPPs to improve the sustainability and efficiency of the energy system.

III. Optimization Algorithm in VPP

The optimization algorithm for VPPs needs a comprehensive approach that not only considers the economic objectives, such as cost minimization and revenue maximization but also integrates technical constraints including battery storage capacities, renewable energy forecasts, and load demand variations.

A critical component of the VPP algorithm involves the formulation of a robust decision-making framework that can efficiently manage the scheduling of appliance usage, the charging and discharging of ESSs, the dispatch of distributed generation units, and the real-time bidding in electricity markets. This requires the inclusion of predictive models that use historical data and real-time inputs to forecast prices, generation capacity, and demand, thereby enabling more accurate and dynamic optimization. The description of the general model proposed in this work aims to support the management of the VPP. The notation used for the model is presented below, together with the mathematical formulation of the objectives, parameters, assumptions, and key constraints of the model.

The VPP operation is modeled using a MILP mathematical programming model that considers integer and continuous variables and all functions, objective and constraints, are linear. In terms of notation for indices and sets, different time periods ($t \in Nt$), different houses ($i \in Ni$) and different appliances ($l \in Nl$) were considered. Nt, Ni and Nl refer to the total number of periods, houses, and appliances respectively. It should be noted that the model in [5] refers to one household, however, as the aim is to study the optimization of a VPP, tests were carried out with several households.

This section presents the mathematical formulation used to model the problem of VPP energy management, which considers the scheduling of shift loads.

Equation 1 represents the objective function of the VPP, which is the calculation of energy costs:

$$\min_{P^{\text{buy}}_{t,i}, P^{\text{sell}}_{t,i}} EC$$

$$EC = \sum_{i=1}^{N_i} \sum_{t=1}^{N_t} \left(P^{\text{buy}}_{t,i} \times tou^{\text{buy}}_{t,i} - P^{\text{sell}}_{t,i} \times fit^{\text{sell}}_{t,i} \right) \times \Delta t + c^{\text{fix}}_{i} \tag{1}$$

where EC represents the energy costs, $P^{\text{buy}}_{t,i}$ is the power purchased from the grid, $tou^{\text{buy}}_{t,i}$ is the time-of-use tariff for buying, $P^{\text{sell}}_{t,i}$ is the power sold to the grid, $fit^{\text{sell}}_{t,i}$ is the tariff for selling power, Δt represents the hourly adjust value, and c^{fix}_{i} is the fixed costs. Equation 2 represents the VPP energy balance:

$$P^{\text{buy}}_{t,i} + P^{\text{gen}}_{t,i} + P^{\text{dch}}_{t,i} = P^{\text{sell}}_{t,i} + \sum_{l=1}^{N_l} P^{\text{shift}}_{t,i,l} + P^{\text{load}}_{t,i} - P^{\text{ch}}_{t,i}, \quad \forall t \in N_t, \forall i \in N_i \tag{2}$$

where $P^{\text{gen}}_{t,i}$ is the power generated by the PV system, $P^{\text{dch}}_{t,i}$ is the power discharged from the battery, $P^{\text{shift}}_{t,i,l}$ is the power of shifted controllable loads, $P^{\text{load}}_{t,i}$ is the power consumed by the uncontrollable loads and $P^{\text{ch}}_{t,i}$ is the power charged to the battery. Equation 3-5 are used to simulate the buying and selling of electricity:

$$0 \leq P^{\text{buy}}_{t,i} \leq \overline{P^{buy}_{t,i}} \times Y^{\text{buy}}_{t,i}, \quad \forall t \in N_t, \forall i \in N_i \tag{3}$$

$$0 \leq P^{\text{sell}}_{t,i} \leq \overline{P^{sell}_{t,i}} \times Y^{\text{sell}}_{t,i}, \quad \forall t \in N_t, \forall i \in N_i \tag{4}$$

$$Y^{\text{buy}}_{t,i} + Y^{\text{sell}}_{t,i} \leq 1, \quad \forall t \in N_t, \forall i \in N_i \tag{5}$$

where $\overline{P^{buy}_{t,i}}$ and $\overline{P^{sell}_{t,i}}$ are the maximum limits for buying and selling electricity, and $Y^{\text{buy}}_{t,i}$, $Y^{\text{sell}}_{t,i}$ are binary variables indicating whether buying or selling actions are taken. In this case, the use of binary variables is important to ensure that only one action is performed in a given period. Equations 6-9 are used to emulate the battery behavior:

$$0 \leq P^{\text{ch}}_{t,i} \leq \overline{P^{ch}_{t,i}} \times Y^{\text{ch}}_{t,i}, \quad \forall t \in N_t, \forall i \in N_i \tag{6}$$

$$0 \leq P^{\text{dch}}_{t,i} \leq \overline{P^{dch}_{t,i}} \times Y^{\text{dch}}_{t,i}, \quad \forall t \in N_t, \forall i \in N_i \tag{7}$$

$$Y^{\text{ch}}_{t,i} + Y^{\text{dch}}_{t,i} \leq 1, \quad \forall t \in N_t, \forall i \in N_i \tag{8}$$

$$\underline{SoC^{bat}_{t,i}} \leq SoC^{bat}_{t,i} \leq \overline{SoC^{bat}_{t,i}}, \quad \forall t \in N_t, \forall i \in N_i \tag{9}$$

where, $\overline{P^{ch}_{t,i}}$ represents the maximum limit for battery charge, $Y^{ch}_{t,i}$ is a binary variable associated to the battery charge action, $\overline{P^{dch}_{t,i}}$ denotes the maximum value for battery discharge, $Y^{dch}_{t,i}$ indicates the binary variable associated to the discharge action, $SoC^{bat}_{t,i}$ represents the state of charge of the battery, $\underline{SoC^{bat}_{t,i}}$ and $\overline{SoC^{bat}_{t,i}}$ are the minimum and maximum limit for the state of charge of the battery. Equation 10 is used to calculate the balance of the battery for period t=1 and equation 11 to calculate the balance in the remaining periods:

$$SoC^{bat}_{t,i} = SoC^{bat\,init}_{i} + (P^{ch}_{t,i} - P^{dch}_{t,i})\Delta t, \quad t = 1, \forall i \in N_i \tag{10}$$

$$SoC^{bat}_{t,i} = SoC^{bat}_{t-1,i} + (P^{ch}_{t,i} - P^{dch}_{t,i})\Delta t, \quad \forall t \in [2, N_t], \forall i \in N_i \tag{11}$$

where $SoC^{bat\,init}_{i}$ is the state of charge value for the first instant. Equation 12 presents the shift power loads calculation:

$$P^{shift}_{t,i,l} = P^{controllable\,load}_{t,i,l} \cdot z_{t,i,l}, \quad \forall t \in N_t, \forall i \in N_i, \forall l \in N_l \tag{12}$$

where, $P^{controllable\,load}_{t,i,l}$ represents the power for the controllable load and $z_{t,i,l}$ indicates whether the load is turned on or off. Equations 13 and 14 are used to control and shift the controllable loads:

$$z_{t,i,l} \leq t^{on}_{t,i,l}, \quad t = 1, \forall i \in N_i, \forall l \in N_l \tag{13}$$

$$z_{t,i,l} \leq z_{t-1,i,l} + t^{on}_{t,i,l}, \quad \forall t \in [2, N_t], \forall i \in N_i, \forall l \in N_l \tag{14}$$

where, $t^{on}_{t,i,l}$ represents a binary variable that indicates the moment when the load is turned on. Equation 15 ensures

that shift loads will be activated during the predefined period number, while equation 16 ensures that each shift load will only be activated once:

$$\sum_{t=1}^{N_t} z_{t,i,l} = P_{t,i,l}^{controllable\,load\,periods}, \quad \forall i \in N_i, \forall l \in N_l \tag{15}$$

$$\sum_{t=1}^{N_t} t_{t,i,l}^{on} = 1, \quad \forall i \in N_i, \forall l \in N_l \tag{16}$$

where $p_{t,i,l}^{controllable\,load\,periods}$ represents the total number of periods that each shift load needs to be activated. Equation 17 presents the limits for binary variables.

$$\{Y_{t,i}^{buy}, Y_{t,i}^{sell}, Y_{t,i}^{ch}, Y_{t,i}^{dch}, z_{t,i,l}, t_{t,i,l}^{on}\} \in \{0,1\} \tag{17}$$

IV. Practical Implementation

The optimization service plays a pivotal role in the operational framework of the system. It is designed as a modular service, which intakes operational data from both external sources, processes this data through a mathematical optimization model, and receive the outputs and directives for the operational control. The core of this service is the optimization algorithm, which relies on solver software to find optimal solutions within a predefined time constraint. Fig. 1 presents the architecture and data workflow of the optimization service.

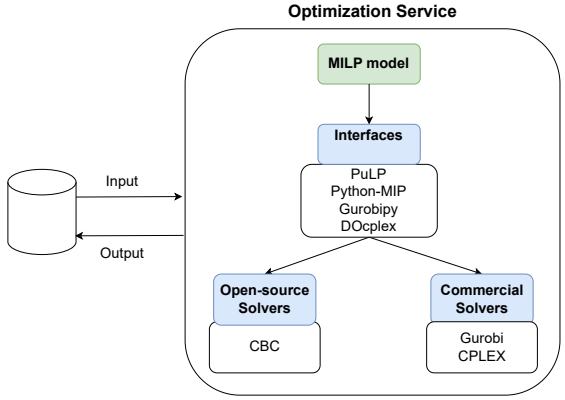

Fig. 1. Architecture and data flow for the optimization service.

The service fetches data through an abstract class, meaning any database system could be integrated if a concrete implementation of the abstract class is provided. This class defines an interface with a single function, which takes the input for the algorithm (defined in Table I) and returns a solution as an output (defined in Table II). The optimization service can be considered as a single worker; i.e., at any given moment, it can only process one single input dataset using one single solver. This does not imply that no parallelization is achievable, however, as it is up to the solvers to adopt such parallel computing strategies internally. If the optimization of multiple input datasets is required, then multiple instances of the optimization service can be deployed, every single one of them with a different solver, if desired. It should be noted that there is no automatic choice of solver; the solver must be chosen manually during the configuration of the service.

The practical implementation of VPP optimization services requires a critical evaluation of the available solver technologies, both commercial and open-source. Our comparison focuses on aspects such as computational efficiency, scalability, ease of integration, and cost-effectiveness. In the realm of optimization problems, especially those as complex and dynamic as those encountered in VPP operations, the choice of solver can significantly impact the efficiency and reliability of the solution process. While open-source solvers offer accessibility and flexibility, commercial solvers typically provide superior performance in terms of computational speed and problem-solving capabilities. Two of the most renowned commercial solvers are Gurobi and CPLEX [16], [17], both of which have established a reputation for their robustness in handling large-scale optimization problems across various industries, including energy management and optimization in VPPs.

To achieve optimal performance, it is important to examine the solver behavior through the log file analysis. If feasible, it is advisable to explore various solvers, in order to expand the range of options. However, it is crucial to exercise caution when relying solely on recommendations for well-established, state-of-the-art solvers, particularly without conducting practical computational experiments. Additionally, it is essential to take into account the complexity of the modeling environment/language and estimate the amount of time required to complete the modeling phase.

A. Commercial Solvers

Commercial solvers play a pivotal role in the efficient handling of complex optimization problems. Below are two solvers, Gurobi and CPLEX, renowned for their advanced algorithms and high-performance capabilities.

- **Gurobi:** Developed by Gurobi Optimization, LLC, the Gurobi Optimizer is a state-of-the-art solver for a wide range of optimization problems, including Linear Programming (LP), MILP, Quadratic Programming, and Mixed-integer Quadratic Programming. Gurobi is celebrated for its high-performance computing capabilities, scalability, and comprehensive support for programming languages and development environments. It integrates advanced algorithms that can be tuned for specific problem types, ensuring optimal performance and accuracy in solving complex optimization models [16].
- **CPLEX:** IBM ILOG CPLEX Optimization Studio encompasses solvers for LP, MILP, QP, and MIQP problems. CPLEX Optimizer stands out for its powerful presolve and cutting-plane algorithms, which efficiently reduce problem size and complexity, significantly speeding up the solution process. Its parallel processing capabilities allow it to leverage multiple processors to handle intricate problems more efficiently, making it particularly suitable

for the demanding optimization tasks associated with managing VPP operations. CPLEX also offers a flexible API and is supported by a wide range of programming languages, facilitating its integration into custom applications [17].

B. Open-Source Solvers

CBC (COIN-OR Branch-and-Cut) is highlighted as a notable open-source solver for addressing various optimization problems [18]. CBC has many key advantages, particularly its cost-effectiveness due to the absence of licensing fees. This attribute is especially valuable for enabling the application of advanced optimization techniques without incurring the high costs associated with commercial software. Such economic efficiency promotes broader adoption and experimentation with optimization models, fostering innovation and research in energy management and beyond. It supports various modeling languages and interfaces, such as PuLP and Python-MIP, facilitating its incorporation into existing systems and workflows. This ease of integration significantly reduces development time and complexity, rendering CBC a practical choice for developers and researchers working on optimization problems. Its compatibility with modeling languages and interfaces ensures that CBC can seamlessly fit into diverse computational environments, enabling the formulation and solving of MILP models with efficiency and precision. Therefore, the following abstraction interfaces were chosen:

- **PuLP:** It is a LP open-source library written in Python. It serves as a modeling layer for LP and MILP problems. The choice of PuLP is motivated by its simplicity in defining decision variables, objectives, and constraints directly in Python, and its compatibility with multiple solvers (CBC, GLPK, CPLEX, and Gurobi), offering flexibility in solver selection based on the problem at hand and available licenses. This makes PuLP ideal for scenarios requiring straightforward modeling and diverse solver option [19].
- **Python-MIP:** Focused on MILP problems, Python-MIP is particularly noted for its performance and advanced features, essential for solving large and complex MILP efficiently. The choice of Python-MIP was driven by the need for a tool that provides deeper integration with MILP solver technologies, enabling more sophisticated problem-solving strategies. This tool provides specialized features for MILP that are not as readily accessible in the other tools [20].

V. Results

This section presents the results from the benchmark study of solvers for VPP optimization, detailing the performance and efficiency of different solver technologies in managing and optimizing VPP operations.

A. Input/Output Data

As detailed in Section III, the model takes in consideration several constraints and decision variables, where only some are provided as input and the rest as output from the solver. The input data used in this study were collected from real historical data of residential energy consumption and local photovoltaic generation. This information was obtained through continuous measurements in households utilizing HEMS. These data include appliance load profiles, photovoltaic energy production, and electricity tariffs [5]. The input data, as outlined in the Table I, includes a variety of parameters essential for the model to perform accurately.

These parameters encompass the number of periods (Nt), the number of houses (Ni), time interval between periods (Δt), and various other technical and economic factors that influence the VPP energy management decisions. The range of values for these parameters is meticulously defined to accommodate the diversity in VPP configurations and operational scenarios.

The output data, presented in Table II, details the results produced by the optimization model, which are critical for making informed decisions regarding energy transactions within the VPP.

These outputs are crucial in strategizing the VPP operations to enhance efficiency, reduce costs, and maintain energy balance.

This distinction between input and output data underscores the comprehensive approach adopted in the optimization model, where various economic and technical aspects are taken into consideration to optimize the VPP operations effectively. The model ability to process a wide range of input parameters and produce actionable outputs facilitates the effective management of energy resources, thereby contributing to the overall efficiency and sustainability of the VPP ecosystem.

B. Testing Environment

All tests were executed on a machine with the following hardware/software specifications:

- **CPU:** Intel Xeon E5-2686 v4 (only 4 vCPU)
- **RAM:** 16 GB
- **OS:** Ubuntu 24.04 LTS (virtualized), running Linux 6.8.0

As for Python and optimization libraries, the following versions were used:

- CPython 3.10.14
- PyPy 7.3.16 (implements the Python 3.10.14 standard)
- Gurobi 11.0.2
- CPLEX 22.1.1.0 (with DOcplex 2.27.239)
- Python-MIP 1.15.0
- PuLP 2.7.0

To measure the scalability of the algorithm, four different test scenarios were considered: 1 house, micro VPP (2 houses), small VPP (4 houses) and community VPP (8 houses). A fixed number of 96 periods with 15 minutes each was set, to allow a full day-ahead optimization. Furthermore, it is important to note that each house has controllable appliances for energy management, more specifically 12 appliances (including, for example, clothes washing machines, hair dryers, coffee makers, and phone chargers, among others). Each

TABLE I
INPUT DATA OF THE ALGORITHM.

Parameter	Designation	Value Range	Unit
N_t	Number of periods	96	-
N_i	Number of houses	$2-8$	-
N_l	Number of controllable loads	12	-
Δt	Time interval between periods	0.25	h
$tou_{t,i}^{buy}$	Price for buying electricity	$0.1034-0.2314$	€/kWh
$fit_{t,i}^{sell}$	Price for selling electricity	0.045	€/kWh
c_i^{fix}	Fixed costs	$0.2197-0.6249$	€
$\overline{P_{t,i}^{buy}}$	Maximum buy amount	$4.6-13.8$	kWh
$\overline{P_{t,i}^{sell}}$	Maximum sell amount	$4.6-13.8$	kWh
$\overline{P_{t,i}^{ch}}$	Maximum charge amount	$0-5$	kWh
$\overline{P_{t,i}^{dch}}$	Maximum discharge amount	$0-5$	kWh
$P_{t,i}^{gen}$	Generated energy forecast	$0-8.474$	kWh
$P_{t,i}^{load}$	Consumed energy forecast	$0.052-9.822$	kWh
$P_{t,i,l}^{controllable\,load}$	Controllable load power	$0.01-5.20$	kWh
$P_{t,i,l}^{controllable\,load\,periods}$	Number of periods that each shift load needs to be activated	$1-8$	-
$SoC_i^{bat\,init}$	Initial state of charge of the battery	$0-1.92$	kWh
$\underline{SoC_{t,i}^{bat}}$	Minimum SoC of the battery	$0-1.824$	kWh
$\overline{SoC_{t,i}^{bat}}$	Maximum SoC of the battery	$0-9.6$	kWh

TABLE II
OUTPUT DATA OF THE ALGORITHM.

Parameter	Designation	Unit
$P_{t,i}^{buy}$	Energy amount to be bought	kWh
$P_{t,i}^{sell}$	Energy amount to be sold	kWh
$P_{t,i}^{ch}$	Energy amount to be charged	kWh
$P_{t,i}^{dch}$	Energy amount to be discharged	kWh
$SoC_{t,i}^{bat}$	Resulting state of charge of the battery	kWh
$P_{t,i,l}^{shift}$	Power of shifted loads	kWh
$Y_{t,i}^{buy}$	Indicates whether energy was bought or not	Binary
$Y_{t,i}^{sell}$	Indicates whether energy was sold or not	Binary
$Y_{t,i}^{ch}$	Indicates whether energy was charged or not	Binary
$Y_{t,i}^{dch}$	Indicates whether energy was discharged or not	Binary
$z_{t,i,l}$	Indicates whether the load is on or not	Binary
$t_{t,i,l}^{on}$	The period when the load is first turned on	Binary

house has the same number of appliances, and the idea is to implement strategies to monitor and schedule appliance usage in a manner that reduces overall energy consumption and minimizes electricity costs, regardless of the appliances themselves.

Each optimization library was tested on each one of the four test scenarios, only accounting for the time spent using the solver (i.e., preparing the problem and optimizing); the fetching of the data and subsequent output were not considered. The data was collected through the measurement of five separate executions on sequentially-running processes.

This study does not aim to provide a direct comparison between PyPy—an alternative Python implementation with a just-in-time (JIT) compiler—and CPython; the two runtimes are merely presented with the aim of analyzing how each optimization library performs in their respective environments. As noted by the PyPy developers, the usage of short-lived processes and external libraries—which is, in essence, the workload featured in this article—is not well-suited for JIT compilation, and, as such, performance benefits will not be visible[1]. In addition, since PyPy has no support for the official Gurobi and DOcplex libraries, those tests were omitted from the final results.

C. Implementation and Discussion

Implementing an optimization service for VPP requires careful consideration of several factors beyond the selection of a solver. These include the complexity of the optimization model, the scalability of the solution (to accommodate varying sizes of VPP networks), and the integration of the optimization service with existing data sources and control systems. The practical experiences of deploying commercial and open-source solvers in VPP optimization tasks have shown that,

[1]https://www.pypy.org/features.html#speed (accessed 24 May 2024)

TABLE III
EXECUTION TIME (IN SECONDS) OF THE SOLVERS, IN TERMS OF HOUSE COUNT.

		House (1)	Micro VPP (2)	Small VPP (4)	Community VPP (8)
Runtime	**Solver**				
	docplex	0.313	1.483	2.543	9.858
	gurobi	0.348	1.314	2.375	4.541
	pulp:cbc	1.174	41.402	118.666	7272.551
CPython	pulp:cplex	0.540	1.868	3.988	20.937
	pulp:gurobi	0.406	1.382	2.674	5.670
	python-mip:cbc	0.798	24.499	110.241	2386.632
	python-mip:gurobi	**0.301**	**1.258**	**2.058**	**4.409**
	pulp:cbc	1.526	41.842	119.127	7279.646
	pulp:cplex	1.002	2.444	4.493	20.965
PyPy	pulp:gurobi	0.872	1.867	3.044	5.512
	python-mip:cbc	1.244	24.979	110.855	2381.551
	python-mip:gurobi	**0.716**	**1.693**	**2.419**	**4.391**

TABLE IV
DETAILED EXECUTION TIME (IN SECONDS) OF THE SOLVERS FOR ONE HOUSE.

Runtime	**Solver**	**Mean ± stdev**	**Min**	**Max**
	docplex	0.313 ± 0.001	0.312	0.314
	gurobi	0.348 ± 0.007	0.339	0.358
	pulp:cbc	1.174 ± 0.007	1.166	1.185
CPython	pulp:cplex	0.540 ± 0.008	0.533	0.552
	pulp:gurobi	0.406 ± 0.003	0.402	0.409
	python-mip:cbc	0.798 ± 0.007	0.791	0.810
	python-mip:gurobi	**0.301 ± 0.003**	**0.297**	**0.305**
	pulp:cbc	1.526 ± 0.013	1.517	1.548
	pulp:cplex	1.002 ± 0.020	0.985	1.031
PyPy	pulp:gurobi	0.872 ± 0.088	0.825	1.029
	python-mip:cbc	1.244 ± 0.008	1.236	1.254
	python-mip:gurobi	**0.716 ± 0.006**	**0.707**	**0.723**

while both types of solvers deliver different performance, their suitability can vary depending on the specific requirements of the task at hand, such as the problem size, the complexity of constraints, and the computational resources available. For example, the computational experiments revealed a significant performance difference between commercial and open-source solvers for MILP problems, even in small examples. This can be attributed to algorithmic optimization, parallel processing, advanced presolve techniques and cutting-edge features. The results can be seen in Table III. The table is split into two sections for CPython and PyPy runtimes.

Observing the previous table, it is also possible to see that certain parameters influence the exploratory tests of different interfaces and solvers:

- **Solver performance variability:** The discussion table highlights significant variability in solver performance, with Gurobi and CPLEX outperforming CBC in terms of speed and reliability, especially as the problem size increases. This aligns with the expectations set in the literature review, where commercial solvers are often recognized for their superior optimization capabilities and efficiency in handling large-scale, complex problems.
- **Impact of problem size on solver efficiency:** As the number of houses increases, the computational complexity of the optimization problem escalates, challenging the solvers capabilities to find optimal solutions within reasonable time frames. The table shows a clear trend of increasing execution time with the number of houses, highlighting the importance of choosing a solver that scales well with problem size for practical VPP applications.
- **Optimization under real-world constraints:** The results underscore the necessity of employing solvers that can effectively handle the intricate constraints typical of VPP operations, such as electricity costs, energy balance and battery management and the management of twelve individual appliances through shift actions. The ability of a solver to navigate these constraints efficiently is critical for optimizing VPP performance, minimizing operational costs, and ensuring energy supply meets demand. Here all interfaces showed the ability to deal with the problem in question.

The results in the table demonstrate that Python-MIP, combined with the Gurobi solver, achieves the best results. This outcome highlights several critical factors in the context of optimization: solver compatibility and efficiency, algorithmic enhancements and leveraging commercial solver strengths.

In order to have a clear understanding of the results, the variation in the performance of the solvers is represented below for an example case of one house. In this sense, Table IV provides a detailed analysis of the execution times for different solvers in a VPP optimization scenario with just one house. Each solver performance is measured in terms of the mean execution time with its standard deviation, as well as the minimum and maximum execution times observed.

In the CPython runtime section, the “python-mip:gurobi” solver exhibits the best average performance with a mean execution time of 0.301 seconds. The “docplex” solver follows closely, indicating that commercial solvers have the upper hand in performance. CBC, both under PuLP and Python-MIP, offers the worst level of performance of the list, with Python-MIP beating PuLP by a slight margin.

Under the PyPy runtime, all solvers follow the same ranking in regard to execution times. Particularly, “python-mip:gurobi” achieves the best performance with a mean execution time of 0.716 seconds.

In order to visually capture the results from Table IV, a box plot visualization (Fig. 2) was generated.

The box plot provides is a graphical tool to represent the variation in observed statistical data using quartiles (e.g., minimum, maximum, and median). In turn, outliers can be plotted as individual points. This visual evidence supports an analysis in terms of consistency and variability in solver performance, providing empirical data on their performance in a VPP context.

In terms of the main observations made, it is worth highlighting that:

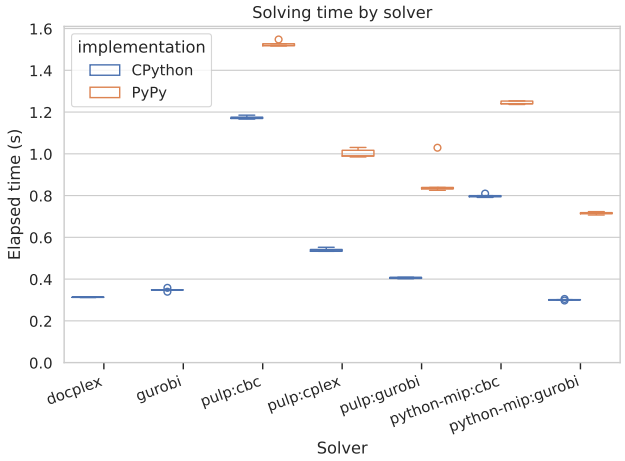

Fig. 2. Box plot with the elapsed time for optimization by various solvers for a scenario with one house setup.

- commercial solvers offer the best level of performance, with low execution times even in more demanding workloads;
- CBC is not suitable for real-time usage, as the Community VPP scenario takes between 30 minutes and 2 hours to complete;
- solvers executed under CPython generally have a lower average solving time compared to those run under PyPy, as the test workload is not suitable for JIT compilation;
- the spread of the data for each solver is quite contained.

This visualization effectively highlights the efficiency and consistency differences between solver implementations and the impact of the Python runtime environment on their performance. These results clearly illustrate that the choice of both solver and runtime can significantly impact performance in VPP optimization tasks. It also indicates that, while open-source solvers like CBC are slower, they may offer a cost-effective alternative to commercial solvers, especially when used for smaller VPP sizes.

In essence, commercial solvers show superior performance, suggesting their suitability for larger and more complex VPP networks. However, the choice of solver also needs to consider factors such as licensing costs, software compatibility, and the availability of technical support. While commercial solvers like Gurobi and CPLEX offer high performance and robust support, their cost might not be justifiable for all project scales. Conversely, open-source solvers like CBC provide a viable alternative with flexibility and customization options, although with potential trade-offs in terms of execution speed and solution optimality. In this sense, it becomes important to discuss the advantages of using open-source solvers. The most evident advantage of using CBC, or any open-source solver, is the absence of licensing fees. This cost efficiency can be a critical factor in enabling the use of advanced optimization techniques without the financial burden associated with commercial software. This democratization of access allows for a wider adoption and experimentation with optimization models, fostering innovation and research in the field of energy management and beyond. Furthermore, open-source software offers unparalleled flexibility and customization opportunities. Users can modify the source code to tailor the solver to their specific needs, optimize its performance for particular types of problems, or even contribute improvements back to the community. This adaptability is particularly beneficial in the rapidly evolving domain of VPPs, where unique and complex optimization challenges can necessitate specialized solver functionalities. In addition, open-source projects benefit from the support of a broad and active community. Users and developers can collaborate, share knowledge, and offer support through forums, repositories, and direct contributions. This collective intelligence can accelerate problem-solving and innovation. Additionally, the transparency of open-source software ensures that the algorithms and methodologies employed are fully visible and open to scrutiny, fostering trust and understanding among users. Finally, the CBC solver, being a part of the Computational Infrastructure for Operations Research (COIN-OR) project, is designed with compatibility and integration in mind. It can be used with various modeling languages and interfaces, such as PuLP and Python-MIP, facilitating its incorporation into existing systems and workflows. This ease of integration can significantly reduce development time and complexity, making it a practical choice for a wide range of optimization problems. It should be emphasized that the purpose was not to observe the effects of the optimization results on the VPPs, but rather to theoretically evaluate the performance of the interfaces.

Thus, for many practical applications, CBC and other open-source solvers can offer sufficient performance and capabilities, especially when the problem is well-structured and falls within the solver optimization strengths. For VPP operators and developers, this suggests a strategic approach to solver and interface selection, taking into consideration not only the mathematical and computational capabilities but also the execution environment. The findings encourage further exploration and benchmarking of different combinations to identify the most effective setup for specific VPP optimization scenarios.

VI. Conclusions and Future Work

The research presented in this document underscores the critical role that solver selection plays in the effective management and optimization of VPPs. Through a detailed comparison of solver performance across various scenarios. The study offers valuable insights that pave the way for the development of more efficient, scalable, and reliable VPP optimization strategies to optimize house energy management considering shifting of electric appliances.

The findings point to Python-MIP and Gurobi as a particularly promising combination for achieving high efficiency in VPP optimization, especially in the context of any sized networks. However, it is crucial for stakeholders to conduct a thorough assessment of a project specific requirements when choosing between commercial and open-source solvers.

Commercial solvers like Gurobi and CPLEX, known for their robustness, appear particularly well-suited for larger, more complex VPP systems, while open-source solvers like CBC are highlighted for their cost efficiency and flexibility, making them ideal for scenarios with small to medium-sized networks where these factors are prioritized. The performance, scalability, and support services of the solver are critical factors that influence the efficiency and reliability of the VPP optimization service.

Despite the strengths of the study, including the comprehensive evaluation of solver performance, there are limitations to consider. Future research should expand on these findings, exploring the scalability of these solutions for larger VPP networks and integrating machine learning techniques to better predict energy demand and production, thereby enhancing the operational efficiency of VPPs. Furthermore, experiments may be carried out that include JIT performance tests as future work. Moreover, meta-heuristic techniques and high-performance computing could be works that leverage the strengths of multiple approaches might provide innovative solutions for VPP optimization in a wider range of scenarios.

REFERENCES

[1] K. O. Adu-Kankam and L. M. Camarinha-Matos, "Renewable energy communities or ecosystems: An analysis of selected cases," *Heliyon*, vol. 8, no. 12, 2022. doi: https://doi.org/10.1016/j.heliyon.2022.e12617

[2] N. Naval and J. M. Yusta, "Virtual power plant models and electricity markets - a review," *Renewable and Sustainable Energy Reviews*, vol. 149, p. 111393, 2021. doi: https://doi.org/10.1016/j.rser.2021.111393

[3] X. Wang, Z. Liu, H. Zhang, Y. Zhao, J. Shi, and H. Ding, "A review on virtual power plant concept, application and challenges," in *2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, 2019. doi: 10.1109/ISGT-Asia.2019.8881433 pp. 4328–4333.

[4] H. M. Rouzbahani, H. Karimipour, and L. Lei, "A review on virtual power plant for energy management," *Sustainable energy technologies and assessments*, vol. 47, p. 101370, 2021. doi: https://doi.org/10.1016/j.seta.2021.101370

[5] R. Faia, P. Faria, and Z. Vale, "Optimizing house energy management with milp considering shifting of electric appliances and battery storage system," in *2023 IEEE PES Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, 2023. doi: 10.1109/ISGTEUROPE56780.2023.10407660 pp. 1–5.

[6] E. A. Bhuiyan, M. Z. Hossain, S. Muyeen, S. R. Fahim, S. K. Sarker, and S. K. Das, "Towards next generation virtual power plant: Technology review and frameworks," *Renewable and Sustainable Energy Reviews*, vol. 150, p. 111358, 2021. doi: https://doi.org/10.1016/j.rser.2021.111358

[7] S. M. Nosratabadi, R.-A. Hooshmand, and E. Gholipour, "A comprehensive review on microgrid and virtual power plant concepts employed for distributed energy resources scheduling in power systems," *Renewable and Sustainable Energy Reviews*, vol. 67, pp. 341–363, 2017. doi: https://doi.org/10.1016/j.rser.2016.09.025

[8] D. Ilic, P. G. Da Silva, S. Karnouskos, and M. Griesemer, "An energy market for trading electricity in smart grid neighbourhoods," in *2012 6th IEEE international conference on digital ecosystems and technologies (DEST)*. IEEE, 2012. doi: 10.1109/DEST.2012.6227918 pp. 1–6.

[9] D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Cañizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, R. Palma-Behnke *et al.*, "Trends in microgrid control," *IEEE Transactions on smart grid*, vol. 5, no. 4, pp. 1905–1919, 2014. doi: 10.1109/TSG.2013.2295514

[10] S. Mashayekh, M. Stadler, G. Cardoso, and M. Heleno, "A mixed integer linear programming approach for optimal der portfolio, sizing, and placement in multi-energy microgrids," *Applied Energy*, vol. 187, pp. 154–168, 2017. doi: https://doi.org/10.1016/j.apenergy.2016.11.020

[11] J. Mohamed, A. Muqbel, A. T. Al-Awami, and I. Elamin, "Optimal demand response bidding and pricing mechanism in distribution network: application for a virtual power plant," in *2018 IEEE Industry Applications Society Annual Meeting (IAS)*. IEEE, 2018. doi: 10.1109/IAS.2018.8544514 pp. 1–8.

[12] G. Weishang, W. Qiang, L. Haiying, and W. Jing, "A trading optimization model for virtual power plants in day-ahead power market considering uncertainties," *Frontiers in Energy Research*, vol. 11, p. 1152717, 2023. doi: https://doi.org/10.3389/fenrg.2023.1152717

[13] Y. Gao, L. Gao, P. Zhang, and Q. Wang, "Two-stage optimization scheduling of virtual power plants considering a user-virtual power plant-equipment alliance game," *Sustainability*, vol. 15, no. 18, p. 13960, 2023. doi: 10.3390/su151813960

[14] G. Ruan, D. Qiu, S. Sivaranjani, A. S. Awad, and G. Strbac, "Data-driven energy management of virtual power plants: A review," *Advances in Applied Energy*, p. 100170, 2024. doi: https://doi.org/10.1016/j.adapen.2024.100170

[15] W. Nafkha-Tayari, S. Ben Elghali, E. Heydarian-Forushani, and M. Benbouzid, "Virtual power plants optimization issue: A comprehensive review on methods, solutions, and prospects," *Energies*, vol. 15, no. 10, p. 3607, 2022. doi: 10.3390/en15103607

[16] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021.

[17] S. Nickel, C. Steinhardt, H. Schlenker, and W. Burkart, *Decision Optimization with IBM ILOG CPLEX Optimization Studio: A Hands-On Introduction to Modeling with the Optimization Programming Language (OPL)*. Springer Nature, 2022.

[18] C. I. O. R. G. (COIN-OR), *CBC (Coin-or Branch and Cut) Solver Documentation*, 2011. [Online]. Available: https://projects.coin-or.org/Cbc

[19] S. Mitchell *et al.*, *PuLP: A Linear Programming Toolkit for Python*, 2005. [Online]. Available: https://coin-or.github.io/pulp/

[20] T. P.-M. Contributors, "Python-MIP: Modeling and solving mixed-integer linear programming problems," 2019, available at https://www.python-mip.com/.