Term paper of Control of Communication and Energy Networks

Decentralized PEV Control Based on a Subgradient Method for Mixed-Integer Programming Problems

Simone Orelli
1749732
DIAG Department
University of Rome "La Sapienza"
orelli.1749732@studenti.uniroma1.it

Antonio Rapuano
2044902
DIAG Department
University of Rome "La Sapienza"
rapuano.2044902@studenti.uniroma1.it

January 2024

Abstract

This term paper presents a decentralized approach to the problem of charging a fleet of plug-in electric vehicles (PEVs) in a smart grid, where global and local constraints and objectives are considered together with the model of the power grid by means of a MPC strategy. The representation of the charging problem involves a combination of continuous and integer variables (specifically of Boolean nature). It is formulated as a mixed-integer linear programming (MILP) problem, which is solved using a subgradient method. The proposed approach is compared with a centralized one, which is typically solved by a branch-and-bound algorithm. The results show that the decentralized approach is able to achieve a performance close to the centralized one, while reducing the computational burden, especially for large fleets of vehicles. These findings are supported by means of simulations performed on a test case.

Contents

1 Introduction

In recent years, the landscape of transportation has witnessed a transformative shift towards electromobility, reflecting a global commitment to sustainable and environmentally conscious practices. This evolution is propelled by advancements in electric vehicle technology, coupled with an increasing societal awareness of the urgent need to reduce carbon emissions in the transport sector. As the number of electric vehicles increases, the demand for efficient charging strategies becomes paramount to ensure seamless integration into the existing infrastructure.

Consider the following scenario: a fleet of PEVs are connected to a charging station. The charging stations are linked to a central controller, which is responsible for managing the charging process. The controller has to decide the charging rate of each vehicle, taking into account their charging requirements and the power limits of the grid connection. The goal is to minimize the total charging cost tracking the aggregated power reference (i.e. desired value for the total power withdrawn from the grid), while ensuring that the charging process is completed according to the preferences of each driver, expressed in terms of final state of charge and charging time.

Typically, the problem is addressed with Model Predictive Control (MPC) techniques, which are well suited for the management of complex systems with multiple constraints. In such instances, the decentralized approach involves dividing the problem into multiple subproblems, which are collaboratively solved by both the plug-in electric vehicles and the control center. This contrasts with the centralized method where data is collected from PEVs, and a singular computation is performed at the control center, which may become impractical for large-scale systems due to the escalating computational burden associated with an increasing number of variables.

Decentralized approaches have been proposed in the past literature. Vujanic et al. (in their work [1]) were among the pioneering researchers exploring scalable PEV charging control solutions by employing decentralized approaches to solve mixed-integer linear problems. Further developments have been proposed by Falsone et al. (in their article [2], for which significant improvements were developed in the recent work [3] by Manieri et al.).

The work presented in this document is a thorough review of the developements proposed by Liberati et al. in their article [4]. Their main contributions lie in modeling the charging process of each PEV as a semi-continuous variable and in the inclusion of the additional objective of tracking a desired aggregated power profile. Here, adopting the solution advanced by Falsone et al. in [2], the authors highlight the differences between a centralized and a decentralized approach, both in terms of problem formulation and solution algorithm. The two approaches are finally compared in terms of computational complexity and performance, by means of simulations.

Table 2 at the end of the document summarizes the nomenclature used.

2 Centralized approach

Consider a set of m PEVs (each with its own specifications) and a MPC prediction horizon of $N = t_f - t_0 + 1$ time steps of duration equal to T. The centralized approach consists in computing the optimal charging/discharging schedule for each PEV p at each time step over the prediction horizon, meeting the preferences of the PEV drivers regarding charging time and final state of charge. It is crucial to ensure that the combined power drawn adheres to specified limits and follows a designated pattern, usually calculated to optimize the influence of recharging activities on the electrical grid. Figure 2.1 shows the block scheme of the centralized approach.

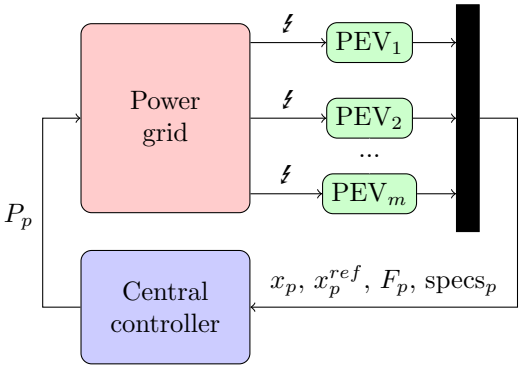

Figure 2.1: Centralized approach block scheme.

2.1 Problem formulation

Goal: compute the optimal charging/discharging schedule for a set of m PEVs connected to the power grid.

PEV charging constraints

$$-P_p^{dis,max} \leq P_{p,i} \leq P_p^{ch,max}, \quad \forall p, i \tag{2.1}$$

$$P_{p,i} = P_{p,i}^{ch} - P_{p,i}^{dis}, \quad \forall p, i \tag{2.2}$$

$$\delta_{p,i}^{ch} P_p^{ch,min} \leq P_{p,i}^{ch} \leq \delta_{p,i}^{ch} P_p^{ch,max}, \quad \forall p, i \tag{2.3}$$

$$\delta_{p,i}^{dis} P_p^{dis,min} \leq P_{p,i}^{dis} \leq \delta_{p,i}^{dis} P_p^{dis,max}, \quad \forall p, i \tag{2.4}$$

$$\delta_{p,i}^{dis} + \delta_{p,i}^{ch} \leq 1, \quad \forall p, i \tag{2.5}$$

Aggregated power constraints

$$P_i = \sum_{p=1}^{m} P_{p,i}, \quad \forall i \tag{2.6}$$

$$P_i \leq P_i^{max}, \quad \forall i \tag{2.7}$$

PEV state of charge evolution
$\forall p, i$:

$$\begin{cases} x_{p,i+1} = x_{p,i} + T(\eta_p^{ch} P_{p,i}^{ch} - \frac{1}{\eta_p^{dis}} P_{p,i}^{dis}) \\ x_{p,t_0} = (x_0)_p \end{cases} \tag{2.8}$$

PEV state of charge constraints

$$x_{p,F_p} = x_p^{ref}, \quad \forall p \tag{2.9}$$

$$x_p^{min} \leq x_{p,i} \leq x_p^{max}, \quad \forall p, i \tag{2.10}$$

Objective function to minimize

$$V = \sum_{i=1}^{N} |P_i - P_i^{ref}| \tag{2.11}$$

that is possible to linearize, introducing the auxiliary variables t_i:

$$V = \sum_{i=1}^{N} t_i \tag{2.12}$$

$$s.t. \quad -t_i \leq P_i - P_i^{ref} \leq t_i, \quad \forall i \tag{2.13}$$

We then introduce a small perturbation term $\xi_{p,i}, \forall p, i$ in V weighting the power to satisfy Assumption 2.4 of [1]. The final form of the objective function is the following:

$$V = \sum_{i=1}^{N} (t_i + \sum_{p=1}^{m} P_{p,i} \xi_{p,i}) \tag{2.14}$$

Charging/discharging schedule
At each iteration of the MPC, the optimal control schedule is computed by solving

$$\begin{bmatrix} P^{ch} \\ P^{dis} \end{bmatrix} = \underset{P^{ch}, P^{dis}}{\arg\min}\, V \tag{2.15}$$

subject to the constraints (2.1)-(2.10), (2.13), (2.14), and using the following notation:

$$P^{ch/dis} \triangleq \begin{bmatrix} P_{1,1}^{ch/dis} & \dots & P_{1,N}^{ch/dis} \\ \vdots & \ddots & \vdots \\ P_{m,1}^{ch/dis} & \dots & P_{m,N}^{ch/dis} \end{bmatrix}$$

The charging/discharging schedule of each PEV is:

$$P_p = \begin{bmatrix} P_{p,1} & \dots & P_{p,N} \end{bmatrix}^T = (P_p^{ch} - P_p^{dis})^T.$$

2.2 Solution algorithm

Algorithm 1: Centralized algorithm.

Input : $\Big\{ N, T, m, (P_i^{max}, P_i^{ref})_{i=1}^N, (P_p^{ch,max}, P_p^{ch,min}, P_p^{dis,max}, P_p^{dis,min}, \eta_p^{ch}, \eta_p^{dis}, (x_0)_p, x_p^{max}, x_p^{min}, x_p^{ref}, F_p)_{p=1}^m, (\xi_{p,i})_{i,p=1}^{N,m} \Big\}$

Output: $\Big\{ (P_{p,i}^{ch}, P_{p,i}^{dis})_{i,p=1}^{N,m} \Big\}$

```
// Solve the optimization
   problem
```

$$\begin{bmatrix} P^{ch} \\ P^{dis} \end{bmatrix} = \underset{P^{ch}, P^{dis}}{\arg\min} \sum_{i=1}^{N} (t_i + \sum_{p=1}^{m} P_{p,i} \xi_{p,i})$$

$$s.t. \quad (2.1) - (2.10), (2.13), (2.14)$$

Notice the large quantity of variables and constraints that the central controller has to deal with.

3 Decentralized approach

The decentralized approach involves the central controller and vehicles collaborating iteratively to find a common solution that meets both local and global constraints and objectives. At each iteration, a tentative charging schedule is computed by each PEV (solving a local optimization problem) and subsequently sent to the main controller. The controller collects the latter and imposes a new set of Lagrange multipliers to the PEVs, corresponding to an updated penalization term for the inner cost functions (see Subsection 3.1). These new multipliers influence the next local tentative charging schedules, and the process iterates until a stop criterion is met. The decentralized approach is summarized in Figure 3.1 (where the iteration process described above is enclosed in the *Fast closed loop*).

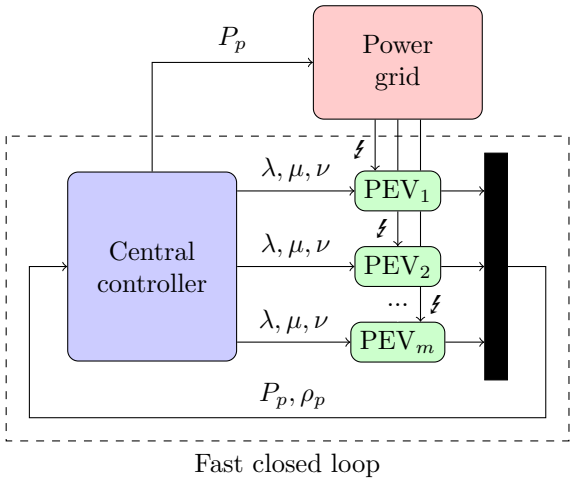

Figure 3.1: Decentralized approach block scheme.

3.1 Problem formulation

Inner minimization problem
The dual function is obtained by dualizing the coupling constraints (2.7) and (2.13) and by imposing $\mu_i + \nu_i \leq 1$, as in [4]:

$$d(\lambda, \mu, \nu) = \sum_{p=1}^{m} min_{P_p} P_p(\xi_p + \lambda + \mu - \nu) + \\ -(\mu - \nu)P^{ref} - \lambda(P^{max} - \rho)$$

subject to the constraints (2.1)-(2.5), (2.8)-(2.10).

The inner minimization problem is decomposed in m decoupled subproblems, one for each PEV p, solvable in parallel. The optimal solution of the p-th subproblem is:

$$P_p = \arg\min_{P_p} P_p(\xi_p + \mu - \nu + \lambda) \qquad (3.1)$$

$$s.t. \quad (2.1) - (2.5), (2.8) - (2.10)$$

Outer minimization problem
Once the optimal solution of the inner minimization problem is found at the level of each PEV, the m tentative charging schedules are shared to the central controller. The controller then updates the Lagrange multipliers accordingly to some decreasing step size $\alpha(k)$, chosen so as to satisfy the following conditions:

$$\lim_{k\to\infty} \alpha(k) = 0,$$
$$\sum_{k=1}^{\infty} \alpha(k) = \infty,$$
$$\sum_{k=1}^{\infty} \alpha(k)^2 < \infty,$$

as requested in the standard dual subgradient method. This process is repeated until the dual subgradient method achieves asymptotic convergence.

3.2 Solution algorithm

Algorithm 2 is a variant of the dual subgradient method, where two are the main steps:

- computation of the subgradient of the dual function (line 5);
- update of the Lagrange multipliers with α as step size (lines 10-13).

The maximum and minimum operations are to be intended component-wise. Notice the decreased number of inputs with respect to the centralized algorithm.

Algorithm 2: Decentralized algorithm.

Input : $\left\{N, T, m, (P_i^{max}, P_i^{ref})_{i=1}^{N}, (\xi_{p,i})_{i,p=1}^{N,m}\right\}$

Output: $\left\{(P_{p,i}^{ch}, P_{p,i}^{dis})_{i,p=1}^{N,m}\right\}$

// Initialize the variables

$k = 0$, $\lambda(k) = 0$, $\mu(k) = 0$, $\nu(k) = 0$

$\overline{s}_p = -\infty$, $\underline{s}_p = +\infty$ for $p = 1, \ldots, m$

// Outer optimization problem

repeat

  for $p = 1, ..., m$ **do**

    // Inner optimization problem

$$P_p(k+1) = \arg\min_{P_p} \Big[P_p(k)\Big(\xi_p + \lambda(k) + \mu(k) - \nu(k)\Big)\Big]$$

$$s.t. \quad (2.1) - (2.5), (2.8) - (2.10)$$

$\overline{s}_p = \max\{\overline{s}_p, P_p(k+1)\}$

$\underline{s}_p = \min\{\underline{s}_p, P_p(k+1)\}$

$\rho_p(k+1) = \overline{s}_p - \underline{s}_p$

  // Update the multipliers

$\rho(k+1) = N \max\left\{\Big(\rho_p(k+1)\Big)_{p=1}^{m}\right\}$

$\lambda(k+1) = max\Big\{0, \lambda(k) + a(k)\Big(\sum_{p=1}^{m} P_p(k+1) - P^{max} + \rho\Big)\Big\}$;

$\mu(k+1) = max\Big\{0, \mu(k) + a(k)\Big(\sum_{p=1}^{m} P_p(k+1) - P^{ref}\Big)\Big\}$;

$\nu(k+1) = max\Big\{0, \nu(k) - a(k)\Big(\sum_{p=1}^{m} P_p(k+1) - P^{ref}\Big)\Big\}$;

  $k++$

until *termination condition is met*;

4 Simulations

The simulations have been performed adopting the parameters in Table 1 using *MATLAB R2023b*. The *Optimization Toolbox* and the *Global Optimization Toolbox* provided the utilities to solve the MILP problems (in particular, the *intlinprog* solver), while *Parallel Computing Toolbox* has been exploited to take avantage of the modularity of the decentralized problem in order to speed up the computation employing multiple CPU cores, each effectively corrisponding to a cluster of PEVs. The code is shown in Appendix A.

Symbol	Value
m	500
$N = F_p$	24
T [min]	20
P^{max} [kW]	$\{500, 200, 500\}$
P^{ref} [kW]	290
$P_p^{ch,max}$ [kW]	3.3
x_p^{max} [kWh]	$[8, 16]$
x_p^{min} [kWh]	1
$(x_0)_p$ [kWh]	$[0.2, 0.5] \cdot x_p^{max}$
x_p^{ref} [kWh]	$[0.55, 0.8] \cdot x_p^{max}$
η_p^{ch}	$[0.925, 0.985]$
ξ_p	$[0, 0.3]$

Table 1: Simulation parameters.

4.1 Centralized approach

Figure 4.1 shows the power profile of the PEVs in the centralized approach. The aggregated power tracks almost perfectly the reference value, without ever exceeding the constraint on the maximum power.

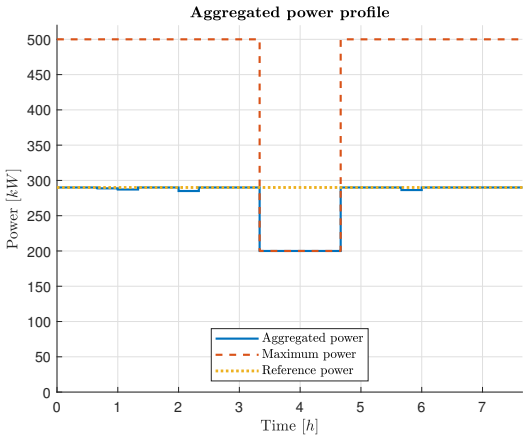

Figure 4.1: Aggregated power profile of the PEVs in the centralized case.

The state of charge evolution of a random batch of 10 PEVs can be seen in Figure 4.2. The state of charge is kept within the limits and the final values reflect the desired ones.

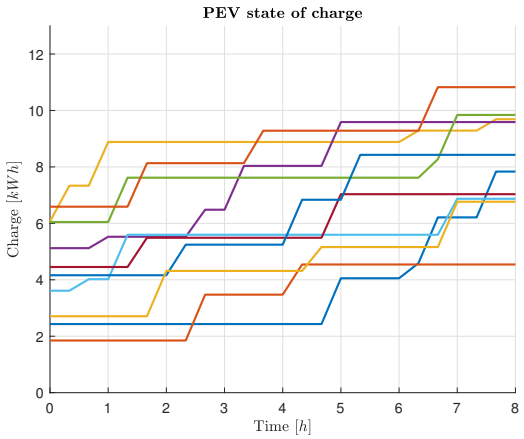

Figure 4.2: State of charge evolution of the PEVs in the centralized case.

Finding the optimal solution of the centralized problem takes about $414.3\,s \simeq 7\,min$. This is typically not acceptable in a real-world scenario, where the optimization problem should be solved in a significantly shorter time than the duration of the sampling step.

4.2 Decentralized approach

The simulation is carried out setting the step size $\alpha(k) = \frac{0.001}{k+1}$. The termination condition is met when the aggregated power profile is under the threshold P_i^{max} and is within $30kW$ of the minimum between P_i^{max} and the reference P_i^{ref}.

Figure 4.3 shows the amount of maximum violation of the global constraint and the maximum deviation between the solution and the reference (regarding the maximum aggregated power). In particular, for small values of k (i.e., the initial iterations of the algorithm) the PEVs try to greedily impose a (globally unfeasible) charging schedule. As the iterations proceed, the central controller modifies the Lagrange multipliers λ, μ, ν in order to force the convergence to a feasible solution, which is reached after 17 iterations. As a matter of fact, that is when the termination condition is met.

The aggregated power profile solution is shown in Figure 4.4. Notice that it always satifies the global constraint, even though it is not as faithful to the reference as the centralized solution. This is due to the fact that the PEVs do not have access to the global information.

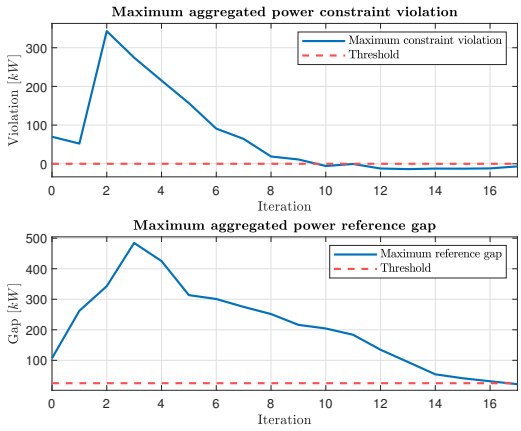

Figure 4.3: Maximum global constraint violation and maximum deviation from the adjusted reference per iteration of the decentralized algorithm.

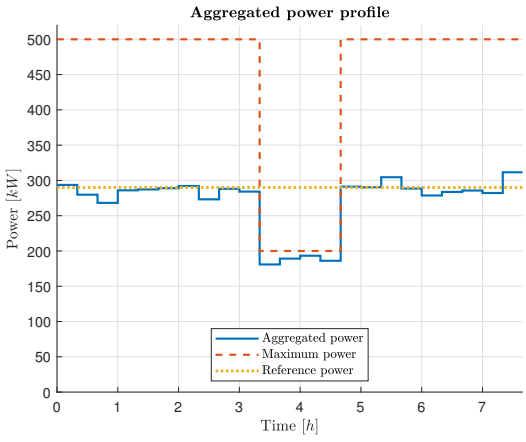

Figure 4.4: Aggregated power profile of the PEVs in the decentralized case.

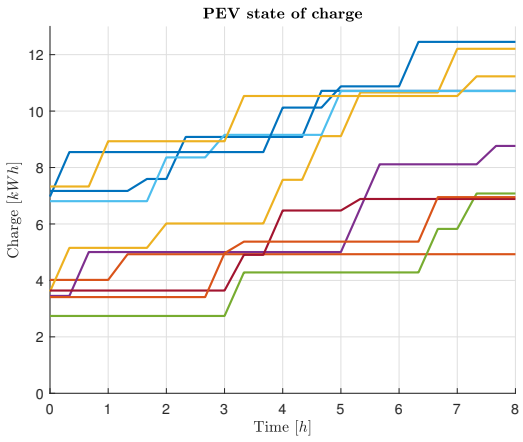

Figure 4.5: State of charge evolution of the PEVs in the decentralized case.

Figure 4.5 displays the state of charge of a random batch of 10 PEVs. As in the centralized case, the charging process is carried out correctly.

The evolution of $||\rho_p||_2$ can be analyzed in Figure 4.6 accordingly to line 9 of Algorithm 2. The increase of the magnitude of ρ_p goes hand in hand with the approach to the globally feasible optimal solution (compare with Figure 4.3).

Finally, in Figure 4.7 the consequent progression of the multipliers is displayed. Iteration after iteration, they are updated according to lines 11-13 of Algorithm 2, and they play a crucial role in the computation of the solutions of the inner optimization problems (line 5). In fact, since in the first iterations the global constraints are violated, the multipliers mutate in order to force the convergence to a feasible solution.

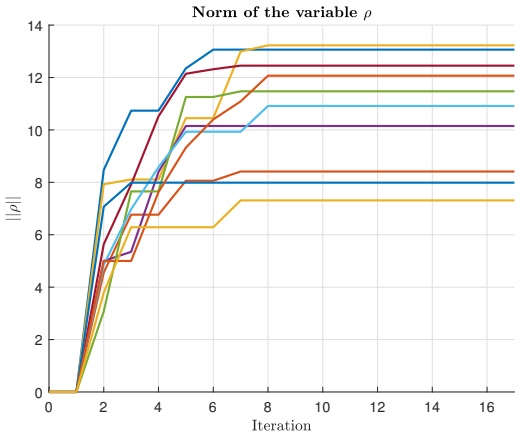

Figure 4.6: Norm of ρ for a random batch of 10 PEVs per iteration of the decentralized algorithm.

In terms of computational burden, there is a significant improvement with respect to the centralized approach. By averaging the time that each PEV takes to solve the inner optimization problem, we get $0.159\,s$. Hence, to carry out 17 iterations, the algorithm takes about $2.7\,s$.

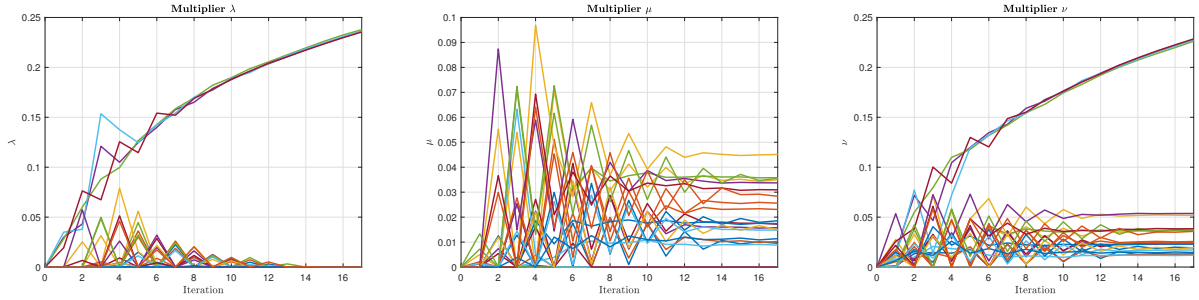

Figure 4.7: Evolution of the Lagrange multipliers per iteration of the decentralized algorithm.

5 Conclusions

In the present document, two different approaches to the PEV charging problem have been presented. The first one is a centralized approach, where the problem is solved by a single entity that has access to all the information about the system. The second one is a decentralized approach, where the problem is solved by a set of agents that have access only to local information, supervised by a central entity. The analysis of the two approaches has been carried out both in qualitative (level of performance) and quantitative (computational burden) terms.

The centralized algorithm manages to find a solution that satisfies the requests of the PEV owners, while keeping the power flow in the network within the limits and almost perfectly equal to the reference. However, it is important to highlight that its computational load is very high, and is not suitable for real-time applications. In fact, the computational time is of the order of minutes, and it is not possible to reduce it significantly without losing the optimality of the solution. Moreover, this controller requires a lot of sensitive information about the PEVs, such as the arrival and departure times, the battery capacity and the required energy.

On the other hand, the decentralized algorithm identifies a feasible solution that is sufficiently close to the centralized one in a noticeably shorter time. Besides, the central unit does not require any sensitive information about the PEVs, as all the necessary data is coded in the form of the local tentative power schedules.

A MATLAB implementation

A.1 Centralized approach

Centralized MPC class

```
function sol = CenMpc(N, T, m, x_min, x_max, x_1, x_ref, i_ref, P_ch_min, P_ch_max, P_dis_min,
    P_dis_max, P_max, P_ref, eta_ch, eta_dis, xi)
    % Define optimization variables
    x = optimvar ('x', m, N+1, 'LowerBound', repmat(x_min, 1, N+1), 'UpperBound', repmat(x_max,
        1, N+1));
    P_ch = optimvar('P_ch', m, N);
    P_dis = optimvar('P_dis', m, N);
    P = optimvar('P', m, N, 'LowerBound', repmat(-P_dis_max, 1, N), 'UpperBound', repmat(P_ch_max
        , 1, N));
    t = optimvar('t', 1, N);
    delta_ch = optimvar('delta_ch', m, N, 'Type', 'integer', 'LowerBound', zeros(m, N), '
        UpperBound', ones(m, N));
    delta_dis = optimvar('delta_dis', m, N, 'Type', 'integer', 'LowerBound', zeros(m, N), '
        UpperBound', ones(m, N));

    % Define optimization constraints
    c1 = P == P_ch-P_dis;
    c2 = P_ch <= delta_ch.*repmat(P_ch_max, 1, N);
    c3 = P_dis <= delta_dis.*repmat(P_dis_max, 1, N);
    c4 = delta_dis+delta_ch <= ones(m, N);
    c5 = delta_ch.*repmat(P_ch_min, 1, N) <= P_ch;
    c6 = delta_dis.*repmat(P_dis_min, 1, N) <= P_dis;
    c7 = x(:, 2:N+1) == x(:, 1:N)+T*(repmat(eta_ch, 1, N).*P_ch(:, 1:N)-P_dis(:, 1:N)./repmat(
        eta_dis, 1, N));
    c8 = x(:, 1) == x_1;
    c9 = optimconstr(m, N);
    for i = 1:m
        c9(i, 1:N-i_ref(i)+2) = x(i, i_ref(i):end) == repmat(x_ref(i), 1, N-i_ref(i)+2);
    end
    c10 = -t <= sum(P, 1)-P_ref;
```

```
    c11 = sum(P, 1)-P_ref <= t;
    c12 = sum(P, 1) <= P_max;

    % Define optimization problem
    prob = optimproblem;
    prob.Objective = sum(t+sum(P.*xi, 1), 2);

    prob.Constraints.c1 = c1;
    prob.Constraints.c2 = c2;
    prob.Constraints.c3 = c3;
    prob.Constraints.c4 = c4;
    prob.Constraints.c5 = c5;
    prob.Constraints.c6 = c6;
    prob.Constraints.c7 = c7;
    prob.Constraints.c8 = c8;
    prob.Constraints.c9 = c9;
    prob.Constraints.c10 = c10;
    prob.Constraints.c11 = c11;
    prob.Constraints.c12 = c12;

    % Solve the optimization problem
    opts = optimoptions(@intlinprog, 'Display', 'iter', 'MaxNodes', 3000);
    sol = solve(prob, 'Options', opts);
end
```

Main script

```
%% Parameters
% Simulation
N = 24; % Prediction horizon
m = 500; % Number of PEVs
T = 1/3; % Time step duration

% Global
P_max_1 = 500*ones(1, 10);
P_max_2 = 200*ones(1, 4);
P_max = [P_max_1, P_max_2, P_max_1]; % Concatenated maximum power for each time step
P_ref = [290*ones(1, N-1), 0]; % Reference power for all time steps

% Local
F = N*ones(m, 1); % Final step for each PEV
P_ch_max = 5*ones(m, 1); % Maximum charging power for each PEV
P_ch_min = 1.3*ones(m, 1); % Minimum charging power for each PEV
P_dis_max = 0*ones(m, 1); % Maximum discharging power for each PEV
P_dis_min = 0*ones(m, 1); % Minimum discharging power for each PEV
x_max = 8*(ones(m, 1)+rand(m, 1)); % Maximum state of charge for each PEV
x_min = 1*ones(m, 1); % Minimum state of charge for each PEV
x_init = (0.2*ones(m, 1)+0.3*rand(m, 1)).*x_max; % Initial state of charge for each PEV
x_ref = (0.55*ones(m, 1)+0.25*rand(m, 1)).*x_max; % Reference state of charge for each PEV
eta_ch = 0.925*ones(m, 1)+0.06*rand(m, 1); % Charging efficiency for each PEV
eta_dis = 1*ones(m, 1); % Discharging efficiency for each PEV
xi = 0.3*rand(m, N); % Random perturbation for each PEV and time step

%% Solution
tic
sol = CenMpc(N, T, m, x_min, x_max, x_init, x_ref, F, P_ch_min, P_ch_max, P_dis_min, P_dis_max,
    P_max, P_ref, eta_ch, eta_dis, xi);
disp("Iteration time: "+toc+" s");
x = sol.x; % State of charge for each PEV at each time step
P = sol.P_ch-sol.P_dis; % Power for each PEV at each time step
```

A.2 Decentralized approach

PEV MPC class

```
% Define a class named PevMpc that models the MPC controller of a PEV (Plug-in Electric Vehicle)
classdef PevMpc
    properties
        % Default parameters for the MPC controller
        N = 24 % Prediction horizon
        T = 1/3 % Time step duration

        x_max = 12 % Maximum state value
        x_min = 1 % Minimum state value
        x_ref = 8.1 % Reference state value

        P_ch_max = 5 % Maximum charging power
        P_ch_min = 1.3 % Minimum charging power
        P_dis_max = 5 % Maximum discharging power
        P_dis_min = 1.3 % Minimum discharging power
        eta_ch = 0.955 % Charging efficiency
        eta_dis = 0.955 % Discharging efficiency
        xi = 0.15*ones(1, 24) % Perturbation factors for the objective function

        s_up = -inf*ones(1, 24) % Upper bounds on power
        s_down = inf*ones(1, 24) % Lower bounds on power
        sol = struct([]) % Solution structure
    end

    methods
        % Constructor for the PevMpc class
        function pevMpc = PevMpc(varargin)
            if(nargin == 12)
                % Set the parameters based on the input arguments
                pevMpc.N = varargin{1};
                pevMpc.T = varargin{2};
                pevMpc.x_max = varargin{3};
                pevMpc.x_min = varargin{4};
                pevMpc.x_ref = varargin{5};
                pevMpc.P_ch_max = varargin{6};
                pevMpc.P_ch_min = varargin{7};
                pevMpc.P_dis_max = varargin{8};
                pevMpc.P_dis_min = varargin{9};
                pevMpc.eta_ch = varargin{10};
                pevMpc.eta_dis = varargin{11};
                pevMpc.xi = varargin{12};
            end
        end

        % Perform one iteration of the MPC algorithm
        function pevMpc = pevMpcIter(pevMpc, x_1, i_ref, lambda, mu, ni)
            % Define optimization variables
            P_ch = optimvar('P_ch', 1, pevMpc.N); % Charging power
            P_dis = optimvar('P_dis', 1, pevMpc.N); % Discharging power
            P = optimvar('P', 1, pevMpc.N, 'LowerBound', -pevMpc.P_dis_max, 'UpperBound', pevMpc.
                P_ch_max); % Net power
            delta_ch = optimvar('delta_ch', 1, pevMpc.N, 'Type', 'integer', 'LowerBound', 0, '
                UpperBound', 1); % Binary variable for charging
            delta_dis = optimvar('delta_dis', 1, pevMpc.N, 'Type', 'integer', 'LowerBound', 0, '
                UpperBound', 1); % Binary variable for discharging
            x = optimvar('x', 1, pevMpc.N+1, 'LowerBound', pevMpc.x_min, 'UpperBound', pevMpc.
                x_max); % State of charge variable
```

```
            % Define constraints
            c1 = P == P_ch-P_dis; % Net power constraint
            c2 = P_ch <= delta_ch*pevMpc.P_ch_max; % Maximum charging power constraint
            c3 = P_dis <= delta_dis*pevMpc.P_dis_max; % Maximum discharging power constraint
            c4 = delta_ch+delta_dis <= 1; % Only one charging or discharging mode can be active
                at a time
            c5 = delta_ch*pevMpc.P_ch_min <= P_ch; % Minimum charging power constraint
            c6 = delta_dis*pevMpc.P_dis_min <= P_dis; % Minimum discharging power constraint
            c7 = x(2:pevMpc.N+1) == x(1:pevMpc.N)+pevMpc.T*(pevMpc.eta_ch*P_ch-P_dis/pevMpc.
                eta_dis); % State update equation
            c8 = x(1) == x_1; % Initial state constraint
            c9 = x(i_ref:end) == pevMpc.x_ref; % Reference state constraint

            % Define the optimization problem
            prob = optimproblem;
            prob.Objective = P*(pevMpc.xi'+mu-ni+lambda); % Objective function
            prob.Constraints.c1 = c1;
            prob.Constraints.c2 = c2;
            prob.Constraints.c3 = c3;
            prob.Constraints.c4 = c4;
            prob.Constraints.c5 = c5;
            prob.Constraints.c6 = c6;
            prob.Constraints.c7 = c7;
            prob.Constraints.c8 = c8;
            prob.Constraints.c9 = c9;

            % Solve the optimization problem
            opts = optimoptions(@intlinprog, 'Display', 'off');
            pevMpc.sol = solve(prob, 'Options', opts); % Store the solution
            pevMpc.s_up = max(pevMpc.s_up, pevMpc.sol.P); % Update the upper bounds on power
            pevMpc.s_down = min(pevMpc.s_down, pevMpc.sol.P); % Update the lower bounds on power
        end
    end
end
```

Main script

```
clear, clc, close all;

%% Parameters
% Simulation
N = 24; % Prediction horizon
m = 500; % Number of PEVs
T = 1/3; % Time step duration
k_max = 100; % Maximum number of iterations

% Global
P_max_1 = 500*ones(1, 10);
P_max_2 = 200*ones(1, 4);
P_max = [P_max_1, P_max_2, P_max_1]; % Concatenated maximum power for each time step
P_ref = [290*ones(1, N-1), 0]; % Power reference for each time step (except the last one)
tol = 25; % Tolerance for power reference deviation

% Local
F = N*ones(m, 1); % Final step for each PEV
x_init = zeros(m, 1); % Initial state of charge for each PEV
pevs(1:m, 1) = PevMpc; % Array of PEV objects
for p = 1:m
    x_max = 8*(1+rand); % Maximum state of charge for each PEV
```

```
    x_init(p) = (0.2+0.3*rand)*x_max; % Initial state of charge for each PEV
    x_ref = (0.55+0.25*rand)*x_max; % State of charge reference for each PEV
    eta_ch = 0.925+0.06*rand; % Charging efficiency for each PEV
    eta_dis = 0.925+0.06*rand; % Discharging efficiency for each PEV
    xi = 0.3*rand(1, N); % Random perturbation for each PEV

    pevs(p, 1) = PevMpc(N, T, x_max, 1, x_ref, 5, 1.3, 0, 0, eta_ch, 1, xi); % Create a PevMpc
        object for each PEV
end

%% Variables
P = zeros(m, N, k_max+1); % Power profile for each PEV at each time step and iteration

rho = zeros(m, N, k_max); % Variable rho for each PEV at each time step and iteration
rho_agg = zeros(N, k_max); % Aggregated variable rho for all PEVs at each time step and iteration
lambda = zeros(N, k_max); % Multiplier lambda for power constraint at each time step and
    iteration
mu = zeros(N, k_max); % Multiplier mu for power reference at each time step and iteration
nu = zeros(N, k_max); % Multiplier nu for power reference at each time step and iteration
alpha = 0.001./((2:k_max)); % Step size for updating multipliers

%% Solution
k = 0; % Iteration counter
while(true)
    k = k+1; % Increment iteration counter
    disp(" ");
    disp("Iteration "+(k-1));

    % Parallel computing
    P_next = zeros(m, N); % Placeholder for next power profile for each PEV
    lambda_curr = lambda(:, k); % Current value of lambda
    mu_curr = mu(:, k); % Current value of mu
    ni_curr = nu(:, k); % Current value of nu
    time = 0; % Total time for parallel computing
    parfor p = 1:m
        tic; % Start timer
        pevs(p) = pevMpcIter(pevs(p), x_init(p), F(p), lambda_curr, mu_curr, ni_curr); % Perform
            MPC iteration for each PEV
        P_next(p, :) = pevs(p).sol.P; % Store the next power profile for each PEV
        time = time+toc; % Accumulate time for parallel computing
    end
    P(:, :, k+1) = P_next; % Update power profiles
    disp("Average iteration time (per PEV): "+(time/m)+" s");
    % ------------------

    P_max_viol = max(sum(P(:, :, k+1), 1)-P_max); % Maximum violation of the maximum aggregated
        power constraint
    P_ref_gap = max(abs(sum(P(:, :, k+1), 1)-min(P_ref, P_max))); % Maximum deviation from the
        adjusted aggregated power reference
    disp("Maximum violation of the maximum aggregated power constraint: "+P_max_viol+" kW");
    disp("Maximum deviation from the adjusted aggregated power reference: "+P_ref_gap+" kW");
    if  (P_max_viol <= 0 && P_ref_gap <= tol) || k == k_max
        P = P(:, :, 1:k+1); % Trim the power profiles to the final iteration
        rho = rho(:, :, 1:k); % Trim the variable rho to the final iteration
        rho_agg = rho_agg(:, 1:k); % Trim the aggregated variable rho to the final iteration
        lambda = lambda(:, 1:k); % Trim the multipliers lambda to the final iteration
        mu = mu(:, 1:k); % Trim the multipliers mu to the final iteration
        nu = nu(:, 1:k); % Trim the multipliers nu to the final iteration
        break; % Exit the loop
    end
```

```
78
79      rho(:, :, k+1) = reshape([pevs.s_up]-[pevs.s_down], m, []); % Update variable rho
80      rho_agg(:, k+1) = N*max(rho(:, :, k+1), [], 1); % Update aggregated variable rho
81      lambda(:, k+1) = max(zeros(N, 1), lambda(:, k)+alpha(k)*(sum(P(:, :, k+1), 1)'-P_max'+rho_agg
            (:, k+1))); % Update multiplier lambda
82      mu(:, k+1) = max(zeros(N, 1), mu(:, k)+alpha(k)*(sum(P(:, :, k+1), 1)'-P_ref')); % Update
            multiplier mu
83      nu(:, k+1) = max(zeros(N, 1), nu(:, k)-alpha(k)*(sum(P(:, :, k+1), 1)'-P_ref')); % Update
            multiplier nu
84  end
```

Symbol	Explanation
m	Number of PEVs.
N	MPC prediction horizon.
t_f	Final instant of the problem.
t_0	Initial instant of the problem.
T	Sampling time.
p	Index to denote a generic PEV.
i	Index to denote a generic time.
$P_{p,i}$	Charging/discharging power of PEV p at time i.
$P_{p,i}^{ch}$	Charging power of PEV p at time i.
$P_{p,i}^{dis}$	Discharging power of PEV p at time i.
$P_p^{ch,min}$	Minimum charging power of PEV p.
$P_p^{ch,max}$	Maximum charging power of PEV p.
$P_p^{dis,min}$	Minimum discharging power of PEV p.
$P_p^{dis,max}$	Maximum discharging power of PEV p.
$\delta_{p,i}^{ch}$	$\in \{0,1\}$. 1 if PEV p is charging at time i, 0 otherwise.
$\delta_{p,i}^{dis}$	$\in \{0,1\}$. 1 if PEV p is discharging at time i, 0 otherwise.
P_i	Aggregated power at time i.
P_i^{max}	Maximum aggregated power at time i.
$x_{p,i}$	State of charge of PEV p at time i.
$(x_0)_p$	Initial state of charge of PEV p.
η_p^{ch}	$\in (0,1)$. Charging efficiency of PEV p.
η_p^{dis}	$\in (0,1)$. Discharging efficiency of PEV p.
F_p	Desired charging end instant of PEV p.
x_p^{ref}	$= x_{p,F_p}$. Desired final state of charge of PEV p.
x^{max_p}	Maximum state of charge of PEV p.
x^{min_p}	Minimum state of charge of PEV p.
P_i^{ref}	Reference aggregated power at time i.
V	Objective function to minimize.
t_i	Auxiliary variable to linearize the objective function.
$\xi_{p,i}$	Perturbation term in the objective function.
λ, μ, ν	Lagrange multipliers.
$\overline{s}_p\, \underline{s}_p$	Auxiliary variables to compute the virtual price term.
ρ	Virtual price term

Table 2: Nomenclature used in the term paper.

References

[1] Robin Vujanic et al. "A decomposition method for large scale MILPs, with performance guarantees and a power system application". In: *Automatica* 67 (2016), pp. 144–156. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2016.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0005109816000078.

[2] Alessandro Falsone, Kostas Margellos, and Maria Prandini. "A decentralized approach to multi-agent MILPs: Finite-time feasibility and performance guarantees". In: *Automatica* 103 (2019), pp. 141–150. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2019.01.009. URL: https://www.sciencedirect.com/science/article/pii/S0005109819300093.

[3] Lucrezia Manieri, Alessandro Falsone, and Maria Prandini. "A novel decentralized approach to large-scale multi-agent MILPs". In: *IFAC-PapersOnLine* 56.2 (2023). 22nd IFAC World Congress, pp. 5919–5924. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2023.10.616. URL: https://www.sciencedirect.com/science/article/pii/S2405896323009874.

[4] Francesco Liberati, Simone Barcellona, and Alessandro Di Giorgio. "Decentralized PEV Control Based on a Subgradient Method for Mixed-Integer Programming Problems". In: *2023 IEEE International Conference on Environment and Electrical Engineering and 2023 IEEE Industrial and Commercial Power Systems Europe (EEEIC / ICPS Europe)*. 2023, pp. 1–6. DOI: 10.1109/EEEIC/ICPSEurope57605.2023.10194623.