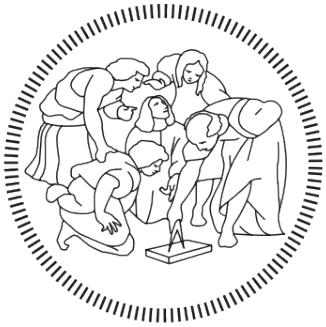


POLITECNICO
MILANO 1863

Advanced Methods for the Optimal Management of the Electrical Grid

Case study 1: "Manual frequency restoration reserve service via prosumers aggregation"

Davide Beretta, Francesco Ferraboli, Davide Zanatta

17 July 2023

Contents

1 Introduction

The increasing penetration of renewable energy sources into the power grid is posing new challenges to grid operators. One of the main challenges is the need to maintain grid frequency stability. This is done in three stages. Primal and secondary regulation are done automatically. If these are not enough, tertiary regulation can be also implemented with manual Frequency Restoration Reserve (mFRR) services, which are a way of offering some degree of flexibility and be willing to provide a power variation over a given time interval when it is manually requested by the Transmission System Operator (TSO).

Traditionally, FRR services have been provided by large conventional power plants. However, as the share of renewable energy sources in the grid increases, there is a growing need for new sources of FRR. Prosumers, who are consumers who also produce their own electricity, can provide a valuable source of FRR. Prosumers have the ability to quickly ramp up or down their power output, which makes them well-suited for providing FRR services. Aggregators become crucial in this scenario because they can optimize the response of a pool of prosumers in order to satisfy a request in the best possible way.

In this report, a simplified problem is tackled, in which renewables are not considered, in order to avoid uncertainties in the production of energy. It is also assumed that the aggregator defines day by day with the TSO the available flexibility reserve and the power reference profile for the next day.

Taking the point of view of the aggregator, in the following we analyze some optimization Mixed-Integer Linear Programming (MILP) models and the performance of both centralized and decentralized algorithms, which become important as the number of aggregated prosumers grows, making centralized computation too complex for standard computational resources.

Part I
Case Study

2 Problem description

We consider an ESP (aggregator) that provides a balancing service to the grid. "It is assumed that the ESP has agreed with the TSO a reference daily power exchange profile and its availability to provide balancing services: if at some point during the day the grid is experiencing an imbalance between production and consumption, the TSO can ask the ESP to modify the reference power profile of the prosumers pool by a certain amount with some tolerance bounds, over a given time frame." [La Bella et al. 2021].

2.1 Variables and parameters

The following table summarizes the variables and the parameters of the problem, together with their constraints.

Table 1: Main optimization and parameters [La Bella et al. 2021]

Symbol	Description	Constraint	Unit
N	Number of prosumers	$N > 0$	integer
M	Number of daily time slots	$M > 0$	integer
t_r	Time slot in which the request arrives	$0 < t_r < M$	integer
t_0	Time slot in which the response starts	$t_r < t_0 < t_f$	integer
t_f	Time slot in which the response stops	$t_0 < t_f < M$	integer
Γ	Power variation requested by the TSO to the prosumers pool	*Unconstrained*	kW
P	Output power of the prosumers pool	*Unconstrained*	kW
P_i	Output power of i-th prosumer	*Unconstrained*	kW
P_i^G	Output power of the controllable generator of i-th prosumer	$P_i^G \geq 0$	kW
δ_i^G	On/Off status of the controllable generator of i-th prosumer	$\delta_i^G \in \{0, 1\}$	boolean
P_i^B	Output power of the battery of i-th prosumer	*Unconstrained*	kW
S_i^B	Energy level of the battery of i-th prosumer	$S_i^B \geq 0$	kWh
P_i^{Pl}	Input power of the programmable load of i-th prosumer	$P_i^{Pl} \geq 0$	kW
δ_i^{PL}	Power consumption level of the programmable load of i-th prosumer	$\delta_i^{PL} \in \{0, 1, 2, 3, 4\}$	integer

3 Problem constraints

$$P(t) = \sum_{i=1}^{N} P_i(t) \tag{1}$$

(1) defines that the output power of the pool as the sum of the aggregated prosumers.

$$P_i(t) = P_i^G(t) + P_i^B(t) - P_i^{Pl}(t) \tag{2}$$

(2) details the composition of the output power of each prosumer.

3.1 Generator

Let now delve into the constraints of the generators.

$$\delta_i^G(t)\underline{P}_i^G \leq P_i^G(t) \leq \delta_i^G(t)\overline{P}_i^G \tag{3}$$

where $\underline{P}_i^G$ and $\overline{P}_i^G$ define the operative range of the generator.

3.2 Battery

$$\underline{P}_i^B \leq P_i^B(t) \leq \overline{P}_i^B \tag{4}$$

where $\underline{P}_i^B$ and $\overline{P}_i^B$ are respectively the minimum and the maximum of the charging/discharging power.

$$\underline{S}_i^B \leq S_i^B(t) \leq \overline{S}_i^B \tag{5}$$

where $\underline{P}_i^B$ and $\overline{P}_i^B$ are respectively the minimum and the maximum energy levels for the battery and $S_i^B(t+1) = S_i^B(0) - \tau_s \sum_{z=0}^{t} P_i^B(s)$. is the battery level at time $t+1$. We use τ_s to indicate the duration of a single timeslot, that is $24h/M$.

This is a very simplified model of the battery, further works may consider other parameters such as charging/discharging efficiency.

3.3 Load

The power of the load is discrete taking a finite set of values.

$$P_i^{Pl}(t) = \frac{\delta_i^{Pl}(t)}{4}\overline{P}_i^{Pl}(t) \tag{6}$$

so it takes values in fractions of the nominal power $\overline{P}_i^{Pl}(t)$

$$\sum_{t=1}^{M} \tau_s P_i^{PL}(t) = E_i^{Pl} \tag{7}$$

so the programmable load of each prosumer must receive the overall amount of energy required for its operation, in other words, E_i^{Pl} is the total energy required by the prosumers' electrical devices. We can shift the load in different time slots, but we have to meet this total energy throughout the day.

3.4 Lead and Rebound Effect avoidance

$$P_i^G(t) = \tilde{P}_i^G(t), \quad t = 1, ..., t_r \tag{8}$$

$$P_i^B(t) = \tilde{P}_i^B(t), \quad t = 1, ..., t_r \tag{9}$$

$$P_i^{Pl}(t) = \tilde{P}_i^{Pl}(t), \quad t = 1, ..., t_r \tag{10}$$

These constraints force the power levels of the various component to follow the reference power profile, defined the day before between prosumer i and the aggregator. We define $\tilde{P}_i(t) = \tilde{P}_i^G(t) + \tilde{P}_i^B(t) - \tilde{P}_i^{Pl}(t)$

$$(1-\epsilon)\Gamma(t) \leq P(t) - \tilde{P}(t) \leq (1+\epsilon)\Gamma(t), t = t_0, ..., t_f \tag{11}$$

where $\tilde{P}(t) = \sum_{i=1}^{N} \tilde{P}_i(t)$ is the agreed profile between ESP and TSO and ϵ is a relative tolerance parameter.

Then we have the rebound effect avoidance constraint, that ensures that the pool returns to the agreed power profile.

$$P_i(t) = \tilde{P_i}(t),\ t = t_f + 1, ..., M \tag{12}$$

(11) and (12) can be rewritten as:

$$(1-\epsilon)\Gamma(t)\delta_{TSO}(t) \leq P(t) - \tilde{P}(t) \leq (1+\epsilon)\Gamma(t)\delta_{TSO}(t), \quad t = 1, ..., M \tag{13}$$

where δ_{TSO} is a row vector of dimension M with ones from position t_0 to position t_f and zeros everywhere else. In other words, it indicates the time in which the request of the TSO is being satisfied.

4 MILP formulation

The idea is to formulate the optimization as a Mixed-Integer Linear Programming problem, namely a problem in which variables are both integer and continuous. In our specific case, an example of continuous variable is P_i^G, while δ_i^G is integer.

We denote as $C_i^G > 0$ the unitary cost of energy produced by the generator G_i, with $C_i^B > 0$ the unitary cost of the aging of the battery G_i and with C_i^{Pl} the unitary cost for changes in the programmable load consumption profile of i-th prosumer.

$$J(.) = \sum_{i=1}^{N}\sum_{t=1}^{M}(C_i^G P_i^G(t) + C_i^B|P_i^B(t) - P_i^B(t-1)| + C_i^{Pl}|P_i^{Pl}(t) - \tilde{P}_i^{Pl}(t)|) \tag{14}$$

4.1 Auxiliary variables

Unfortunately, this cost function is non-linear, because it contains absolute values. To make it linear, we introduce two auxiliary variables:

$$h_i^B(t) = |P_i^B(t) - P_i^B(t-1)| \tag{15}$$

and

$$h_i^{Pl}(t) = |P_i^{Pl}(t) - \tilde{P}_i^{Pl}(t)| \tag{16}$$

Then we modify (15) and (16) to become a systems of linear inequalities:

$$\begin{aligned} P_i^B(t) - P_i^B(t-1) \leq h_i^B(t), \\ P_i^B(t-1) - P_i^B(t) \leq h_i^B(t) \end{aligned} \tag{17}$$

$$\begin{aligned} P_i^{Pl}(t) - \tilde{P}_i^{Pl}(t) \leq h_i^{Pl}(t), \\ \tilde{P}_i^{Pl}(t) - P_i^{Pl}(t) \leq h_i^{Pl}(t) \end{aligned} \tag{18}$$

4.2 Linear cost function

So, the cost function is made linear as:

$$J(.) = \sum_{i=1}^{N}\sum_{t=1}^{M}(C_i^G P_i^G(t) + C_i^B h_i^B(t) + C_i^{Pl} h_i^{Pl}(t)) \tag{19}$$

In the following, we define $x_i = [P_i^G(0)...P_i^G(M)\ \ \delta_i^G(0)...\delta_i^G(M)\ \ P_i^B(0)...P_i^B(M)\ \ P_i^{Pl}(0)...P_i^{Pl}(M)$ $\delta_i^{Pl}(0)...\delta_i^{Pl}(M)\ \ h_i^B(0)...h_i^B(M)\ \ h_i^{Pl}(0)...h_i^{Pl}(M)]^T$ as the vector of the local decision variables for prosumer i, for each $i = 1...N$ and $c_i = [C_i^G...C_i^G\ \ 0...0\ \ 0...0\ \ 0...0\ \ 0...0\ \ C_i^B...C_i^B\ \ C_i^B...C_i^B]^T$ the $(7M) \times 1$ vector containing the costs of prosumer i, for each $i = 1...N$ associated to the variables, for each time slot.

We can then write our optimization problem in a compact way:

$$\begin{aligned} \min_{x1,...,x_N} \quad & \textstyle\sum_{i=1}^{N} c_i^T x_i \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{N} A_i x_i \leq b \\ & x_i \in X_i, \qquad i = 1...N \end{aligned} \tag{20}$$

In this formulation, $\sum_{i=1}^{N} A_i x_i \leq b$ summarizes the coupling constraint in (13), whereas X_i is the set that comprises all the constraints from (3) to (10), together with (17) and (18).

5 Optimization

5.1 Centralized approach

In the centralized approach, the optimization is done entirely by a central computing unit that optimizes for all the prosumers at once. Obviously, this approach is itself time consuming and it becomes more and more computationally prohibitive as the size grows. A centralized solution needs that each agent, in this case each prosumer, communicates to the aggregator the values of their decision variables, and this may cause privacy-related issues, as in scenarios like this, a prosumer might not want to disclose publicly their own consumption habits or costs.

In order to tackle this kind of problems, proposals have been suggested for decentralized approach, sacrificing optimality for computational feasibility. Decentralized methods require agents to communicate only their contribution $A_i x_i$, i.e. their power profile $P_i(t)$ and that solves also privacy issues.

5.2 Decentralized approaches

In decentralized approaches, instead, we exploit dual decomposition, so that agents solve locally and potentially in parallel a simplified optimization problem, then they communicate to the aggregator only the solution of the problem, namely the power profile. Upon receiving all the agents' profiles, the central unit enforces the coupling constraint, by updating the dual variable λ. With this formulation, the dual problem is convex, even if the primal is not.

5.2.1 Resource tightening

We have that each agent minimizes its own optimization problem minimizing $c_i^T x_i + \lambda^T A_i x_i$ (from the dual formulation)[Falsone, Margellos, and Prandini 2019, Vujanic et al. 2016], subject to $x_i \in X_i$, then they communicate $A_i x_i$ to the aggregator that sums all the contributions and subtracts b to obtain the violation of the coupling constraint. If the violation is zero or less, then the proposed solution is feasible and the algorithm returns, otherwise, the aggregator updates the Lagrange multiplier λ to increase the weight applied to A_i in the local problem, so that agents are forced to find a new solution that is supposed to reduce the violation. λ increases at each iteration, until it converges to a stationary value. When this happens, we apply resource tightening, by increasing a parameter, ρ that influences the dynamics of λ, and the process is repeated. This method is powerful, but it has a drawback: it can make the problem unfeasible if ρ grows too much.

The main difference between the two algorithms that we propose lies in how this resource tightening is handled, in particular, in how the parameter ρ is updated upon convergence of λ.

5.2.2 Vujanic et al. 2016

We will refer to this first approach (1) as Vujanic algorithm. It is a variant of dual subgradient algorithm, and it considers the worst case tightening.
$\tilde{\rho} = p \ \max_i\{\rho_i\}$. p is the dimension of the resource, while each ρ_i is computed as the difference between the maximum and the minimum values of the power profile of agent i. In this case, if ρ is growing too much we simply don't update it.

5.2.3 Falsone

The idea of this second approach (2) is to obtain a solution that is closer to optimality. It is inspired by the previous one, but it is less conservative because ρ is adaptive instead of worst case resource tightening. We update ρ by adding the minimum among the h-infinity norm of the violations of the last w inner loop iterations. If ρ is too big, then we restart the algorithm resetting all the parameters and relaxing the constraint in the error tolerance ϵ with respect to the request of the TSO. We will call this algorithm Falsone.

For both algorithms, ρ is not **ok** if $\rho^{ub}(t) + \rho^{lb}(t) \geq 2\epsilon\Gamma(t),\ t = t_0, ..., t_f$, where ρ^{ub} and ρ^{lb} are the components of ρ associated with the right and left constraints in (11) respectively.[La Bella et al. 2021]

Algorithm 1 Vujanic algorithm [Falsone, Margellos, and Prandini 2019]

$\lambda = 0$
$\rho(0) = 0$
$\overline{s}_i(1) = -\inf,\ i = 1...N$
$\underline{s}_i(1) = +\inf,\ i = 1...N$
$k_{out} = 1$ ▷ we use 1 because Matlab indexing starts from one
while $v(k_in) > 0$ and $k_o < 15$ **do**
  $k_{in} = 1$ ▷ inner loop index
  while not(λ converged) and $v(k_{in}) > 0$ **do**
    for $i = 1$ to N **do**
      $x_i(k_{in}+1) \leftarrow \arg\min_{x_i \in X_i}(c_i^T + \lambda(k_{in})^T A_i)x_i$
    end for
    $v(k_{in}+1) = \sum_{i=1}^{N} A_i x_i(k_{in}+1) - b$
    $\lambda(k_{in}+1) = [\lambda(k_{in}) + \alpha(k_{in})(v(k_{in}) + \rho(k_{out}))]_+$
    $k_{in} = k_{in} + 1$
  end while
  $\overline{s}_i(k_{out}+1) = \max\{\overline{s}_i(k_{out}), A_i x_i(k_{in}+1)\},\ i = 1...N$
  $\underline{s}_i(k_{out}+1) = \min\{\underline{s}_i(k_{out}), A_i x_i(k_{in}+1)\},\ i = 1...N$
  $\rho_i(k_{out}+1) = \overline{s}_i(k_{out}+1) - \underline{s}_i(k_{out}+1),\ i = 1...N$
  $\rho(k_{out}+1) = p\ \max\{\rho_1(k_{out}+1), ..., \rho_n(k_{out}+1)\}$
  if ρ is not **ok then**
    $\rho(k_{out}+1) = \rho(k_{out})$ ▷ We simply don't update ρ
  end if
  $\lambda(1) = \lambda(k_{int})$
  $k_{out} = k_{out} + 1$
end while

Algorithm 2 Falsone algorithm [La Bella et al. 2021]

$v(1) = +\inf$
while $v(k_{in}) > 0$ **do**
  $\lambda = 0$
  $\rho(0) = 0$
  $k_{out} = 1$
  while $v(k_{in}) > 0$ and $k_o < 15$ and ρ is **ok do**
    $k_{in} = 1$
    while not(λ converged) and $v(k_{in}) > 0$ **do**
      for $i = 1$ to N **do**
        $x_i(k_{in}+1) \leftarrow \arg\min_{x_i \in X_i}(c_i^T + \lambda(k_{in})^T A_i)x_i$
      end for
      $v(k_{in}+1) = \sum_{i=1}^{N} A_i x_i(k_{in}+1) - b$
      $\lambda(k_{in}+1) = [\lambda(k_{in}) + \alpha(k_{in})(v(k_{in}) + \rho(k_{out}))]_+$
      $k_{in} = k_{in} + 1$
    end while
    $\overline{k}_{in} = \arg\min_{k > k_{in} - w}(||[v(k)]_+||_{\inf})$
    $\rho(k_{out}+1) = \rho(k_{out}) + [v(\overline{k}_{in})]_+$
    $\lambda(1) = \lambda(k_{int})$
    $k_{out} = k_{out} + 1$
  end while
  if ρ is not **ok** or $k_{out} \geq 15$ **then**
    $\epsilon = 1.5\epsilon$ ▷ We restart the algorithm increasing the tolerance with respect to the TSO request
    update b ▷ And we update the constraint accordingly
  end if
end while

6 Simulation and results

6.1 Simulation setup

To test our algorithms we performed a series of simulations in matlab. We run our program on a PC equipped with a Ryzen5 1600 processor with 6 cores and 12 threads, with base clock of $3.20GHz$ and $16GB$ of RAM DDR4.

The programmable load, the generator and the battery were dimensioned following what has been done in La Bella et al. 2021. We set up our simulations having the reference profiles generated at random for each run (making sure that the constraints of the single prosumers were satisfied), the TSO request arriving at $6:45$ and the response starting at $7:00$ until $9:00$. The amplitude of the request was 18 times the number of prosumers, with an initial tolerance factor $\epsilon = 0.05$.

We decided to simulate pools of prosumers with different sizes, varying from 60 to 3000 prosumers. When running the decentralized algorithms, we used the **parfor** function of the *Matlab parallel computing toolbox* so that each prosumer performed the local optimization using only a thread of the CPU.

We defined the step-size parameter $\alpha(k) = \frac{\alpha_1}{(k+1)^{\alpha_2}}$ we used $\alpha_1 = 0.0035/N$, $\alpha_2 = 0.51$ in order to have it satisfy $\sum_{k=1}^{\infty} \alpha(k) = \infty$ and $\sum_{k=1}^{\infty} \alpha(k)^2 < \infty$.

We consider λ as converged, if in the last 7 iterations, one of the following is verified Manieri, Falsone, and Prandini Not yet published:

$$\| \frac{\lambda(k+1) - \lambda(k)}{\lambda(k)} \|_\infty < \Theta_{rel} \tag{21}$$

$$\| \lambda(k+1) - \lambda(k) \|_\infty < \Theta_{abs} \tag{22}$$

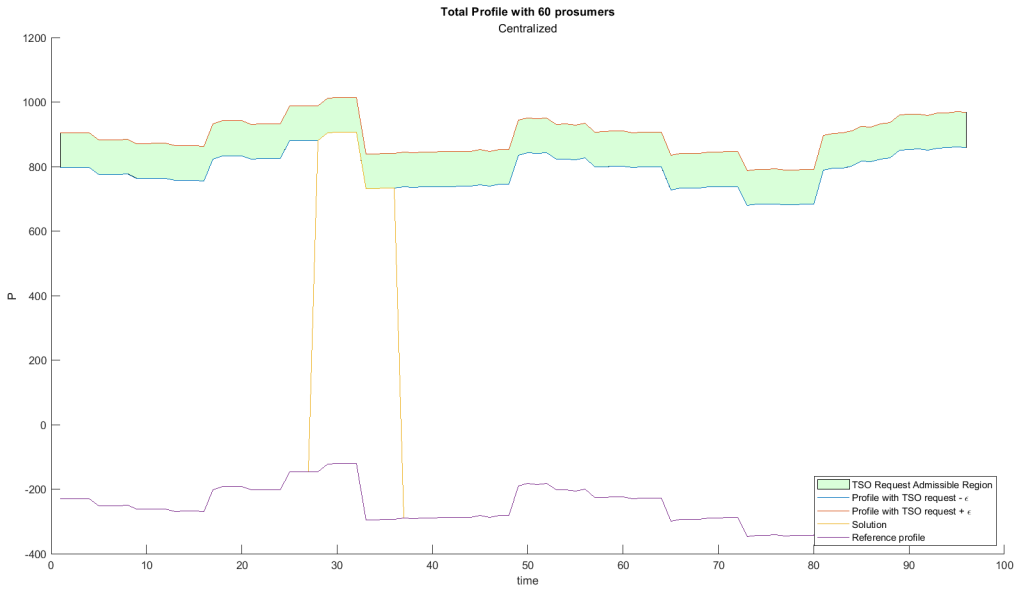
Where the ratio of two vectors has to be intended component-wise, and the absolute and relative tolerances are set to $\Theta_{abs} = 10^{-4}$, $\Theta_{rel} = 10^{-3}$.

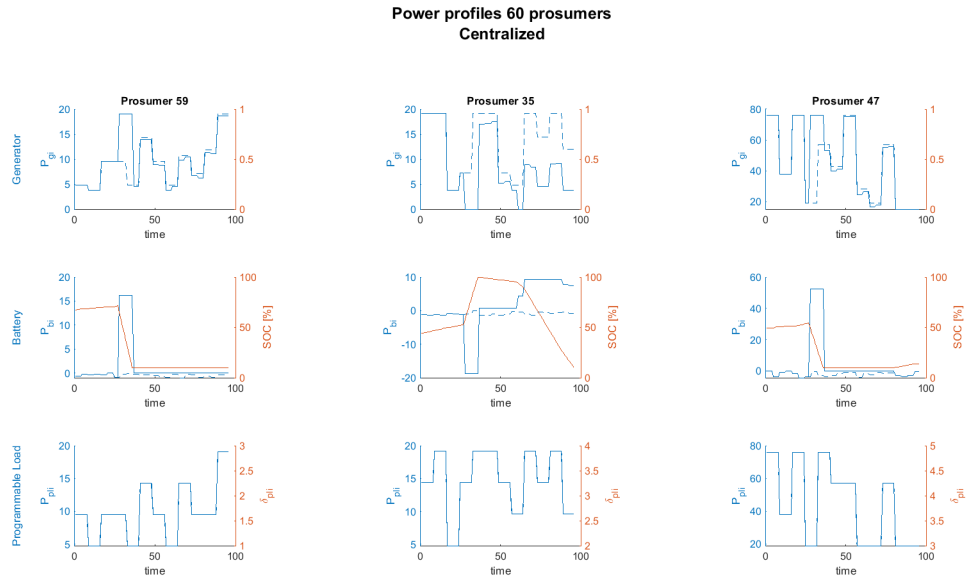
6.2 Results

We propose here an example of the performance of the algorithms with a pool of 100 prosumers.

6.2.1 Centralized

Starting from the centralized algorithm, the first picture below shows the profile of the whole pool, while the second shows the profile of three random prosumers. In the latter, the power profiles are shown in blue, with the solutions shown with solid lines and the reference profile with dashed lines.

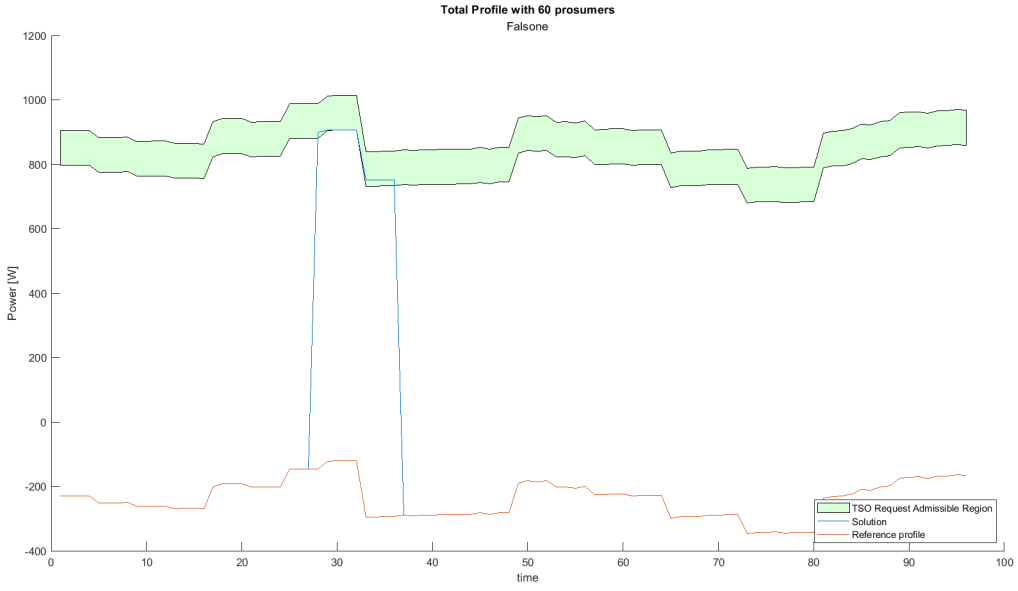




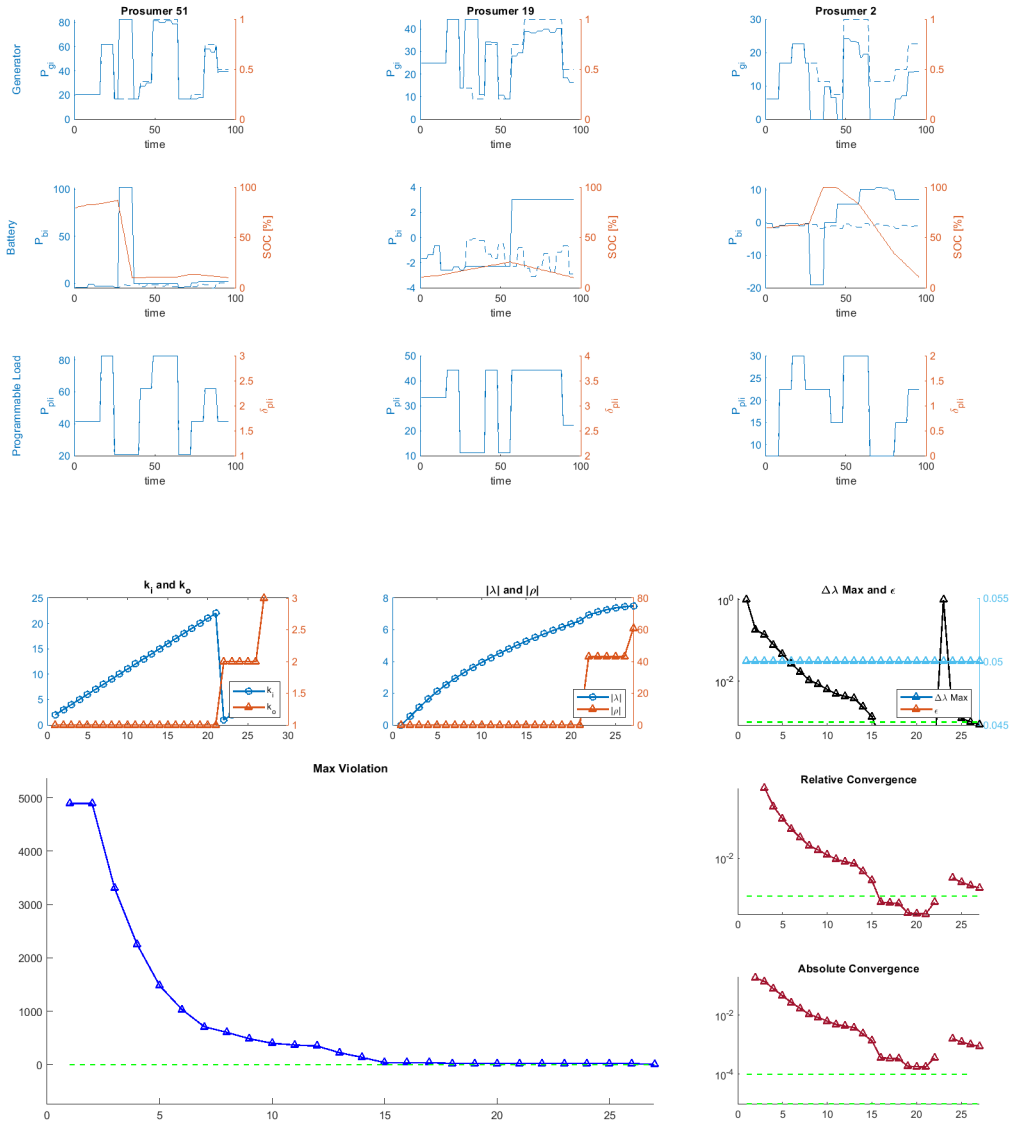
6.2.2 Falsone

Here are shown the graphs for Falsone algorithm (2). The first two graphs are similar to the ones proposed for the centralized, while the last one shows the solution progress, through the iterations of the algorithm.

It is possible to note that the solution found by this algorithm is different from the one of the centralized algorithm, because theoretically we should sacrifice optimality to achieve faster computation (as we discuss later).



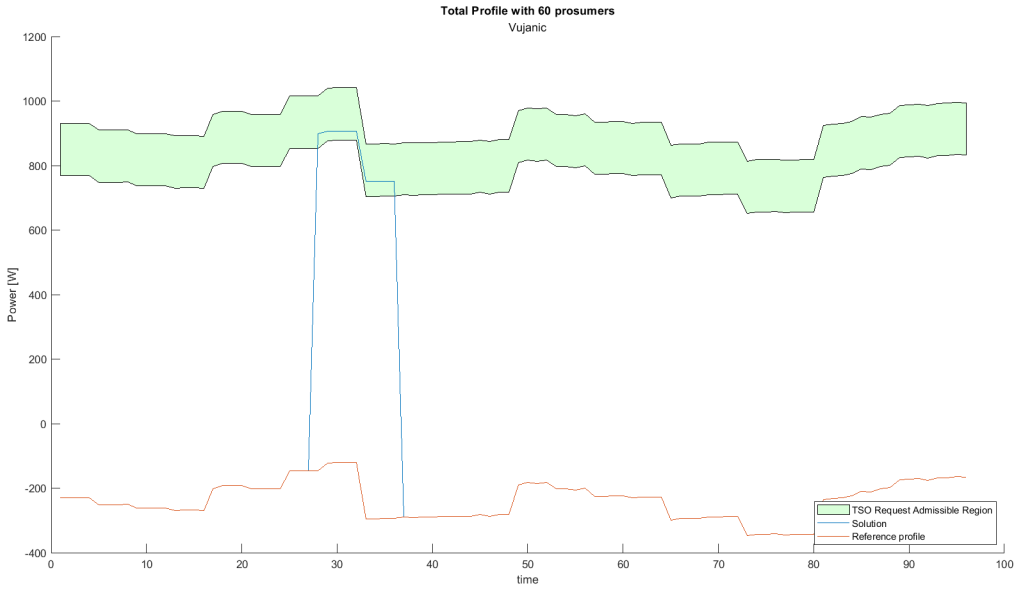
Power profiles 60 prosumers
Falsone et al

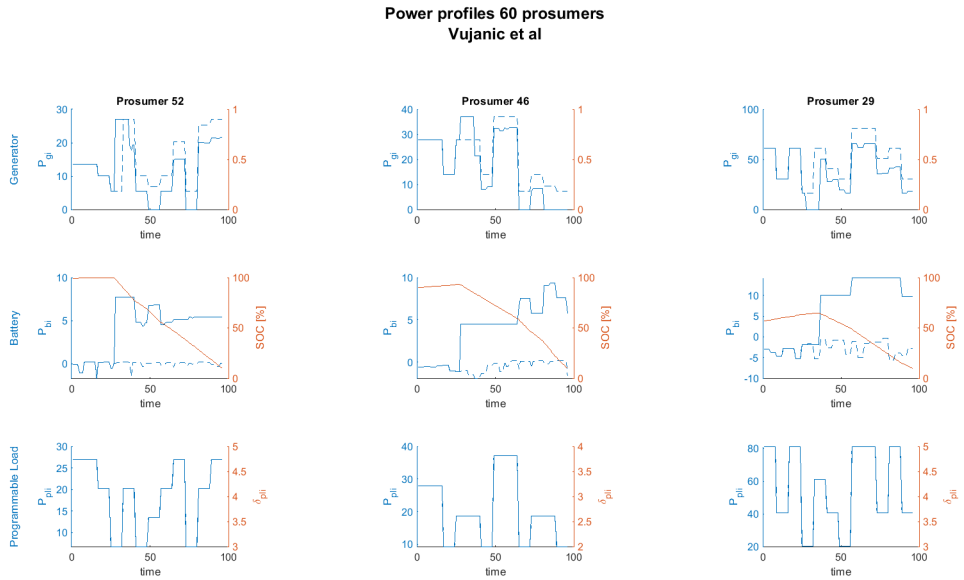


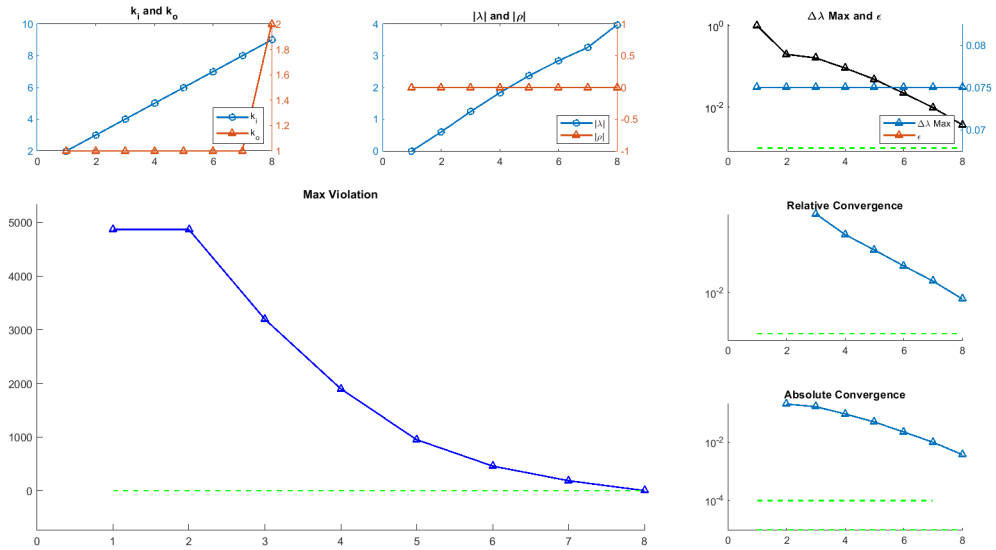
6.2.3 Vujanic

Lastly, we present the graphs for Vujanic algorithm (1), that are analogous to those of Falsone.

This time, the green area enlightening the admissible region for the total profile during the TSO request is wider, since ϵ was incremented to reach the solution. Observing the other graphs in appendix A, we can note that this behaviour is quite frequent using Vujanic algorithm.







6.2.4 Comparison

The first comparison we made is between the execution time of each algorithm. We first computed the mean execution time of the single prosumer in the pool and then we multiplied it by the total number of inner iterations of the algorithm. We did not report the time for the centralized algorithm, since it always exceeds the time threshold of 7200s (which is much bigger than any of the execution time recorded for the decentralized cases) and hence it stopped without reaching the optimal solution.

Table 2: Time comparison

	Falsone				Vujanic			
N	epsilon	Inner iter.	Outer iter.	Execution time	epsilon	Inner iter.	Outer iter.	Execution Time
60	0.05	27	3	113.54s	0.075	8	2	64.56s
70	0.05	65	4	162.57s	0.37969	14	2	33.78s
80	0.05	66	4	151.02s	0.16875	29	2	65.62s
90	0.05	41	4	95.64s	1.9222	27	2	62.95s
100	0.05	34	2	79.47s	0.05	34	2	78.54s
150	0.05	42	3	102.60s	1.9222	45	4	106.96s
250	0.05	92	4	203.99s	0.37969	31	2	71.74s
500	0.05	45	3	101.98s	0.8543	101	4	220.75s
1000	0.05	27	3	61.65s	0.37969	38	4	87.41s
2000	0.05	35	3	82.51s	0.25313	44	4	103.40s
3000	0.05	35	3	80.61s	0.16875	44	4	99.04s
4000	0.05	34	3	78.66s	0.075	41	4	101.70s
5000	0.05	32	3	75.78s	0.075	41	4	97.02s

The first observation we can make is that Falsone finds a solution keeping ϵ at its initial value, while, for Vujanic, this does not happen. In order to find a solution it is necessary to increment ϵ, sometimes even reaching meaningless values.

Given this fact, trying to make any other comparison is difficult, because we are dealing with totally different optimizations. Moreover, the centralized algorithm could not find an optimal solution, hence we could not compute a real value for the optimality gap.

The only value that we were able to compute was the estimated bound $\overline{\Delta J}_{\%}$, as discussed in La Bella et al. 2021. We just did for 100 prosumers, the only number for which a comparison makes sense, because Vujanic reached a solution within the initial tolerance.

In particular, for 100 prosumers, the estimated bound for both algorithms is $\overline{\Delta J}_{\%} = 0.2\%$, that is reasonable. As shown in the table, also the times are comparable, because Vujanic takes about one second less than Falsone to find a solution.

For completeness, below are the $\overline{\Delta J}_{\%}$ values of the Falsone algorithm.

Table 3: $\overline{\Delta J}_{\%}$

N	Estimated optimality gap bound [%]
60	0.0877
70	2.76
80	0.9251
90	13.3059
100	0.2
150	13.7972
250	2.7043
500	5.9698
1000	2.6777
2000	1.8743
3000	1.129
4000	0.4319
5000	0.3737

7 Conclusions

In conclusion, after the analysis of the proposed algorithms for the task at hand, we can say that with our settings, the decentralized proposals are significantly faster than the centralized one, especially with pools of larger dimension. Thus, for time-sensitive applications, in which we want to obtain a solution in a small time slot, we can sacrifice optimality, and prefer a decentralized algorithm.

Among the proposed solutions, the algorithm that we called Falsone (2), proposed in La Bella et al. 2021, works better than the other, that we called Vujanic (1), proposed in Vujanic et al. 2016. This is mainly because Falsone algorithm is able to find the solution staying in the desired tolerance region, while Vujanic needs a wider tolerance to do it.

Part II
Renewable extension

8 Renewables

Since we are moving towards decarbonization of the electrical grid, it would also be interesting to apply the same methods to a grid in which there are non-programmable power generators such as photovoltaic plants.

In order to introduce the PV generators into the problem, we had to modify a little the optimization problem.

- we added a new variable, $PVc_i \in [0,1]$, representing the curtailment factor of i-th prosumer's PV generator, which had a reference generation profile PV_i^{ref}, randomly generated, following a realistic PV profile (we considered also the presence of clouds). Obviously, knowing in advance the exact power profile of the PV generator is a strong assumption. One may make some forecasting and knowing the error use the worst case scenario to perform this optimization, but this was out of the scope of this case study, so we used an exact profile.
- we decided to remove the programmable generator for the prosumers which had a PV generator
- we set the maximum power of the battery equal to the maximum power provided by the PV plant (which is also equal to the maximum power required by the load).
- we introduced a cost for the wasted PV energy that we called C_i^{PVc}

Similarly to what we did before in the equations (15) and (16), we had to insert an auxiliary variable

$$h_i^{PV}(t) = (1 - PVc_i(t))PV_i^{ref}(t) \tag{23}$$

That we decomposed in two inequalities

$$(1 - PVc_i(t))PV_i^{ref}(t) \leq h_i^{PV}(t) \tag{24}$$

$$-(1 - PVc_i(t))PV_i^{ref}(t) \leq -h_i^{PV}(t) \tag{25}$$

Thus, the cost function became

$$J(.) = \sum_{i=1}^{N}\sum_{t=1}^{M}(C_i^G P_i^G(t) + C_i^B h_i^B(t) + C_i^{Pl} h_i^{Pl}(t) + C_i^{PVc} PV_i^{ref}(t) PVc_i(t)) \tag{26}$$

Also, we added a lead and rebound effect avoidance for PV

$$h_i^{PV}(t) = PV_i^{ref}(t), \quad t = 1, ..., t_r \tag{27}$$

and

$$P_i(t) = P_i^G(t) + h_i^{PV}(t) + P_i^B(t) - P_i^{Pl}(t) \tag{28}$$

$$\tilde{P}_i(t) = \tilde{P}_i^G(t) + PV_i^{ref}(t) + \tilde{P}_i^B(t) - \tilde{P}_i^{Pl}(t) \tag{29}$$

We followed the same approach as the one of the case study to apply the centralized and the Falsone algorithms. We decided to ignore Vujanic algorithm in this case, since, as discussed above, it does not find the solution within the desired tolerance region in the majority of the cases.

8.1 Simulation and results

8.1.1 Simulation setup

For each pull of prosumers we tried to solve the problem with the same loads and with a varying percentage of PV plants penetration in the pool, namely 0%, 50%, 75%, 90% and 100%. The other setups were the same as for the simulation without renewables.

8.1.2 Results and observations

By performing some tests, we found out that sometimes that the problem becomes unfeasible, especially with higher penetration of PV power production in the pool ($> 50\%$) and with positive values of Γ. This is probably because the power generation is not programmable, hence we have to rely on the PV profile to satisfy the TSO request, and this is not always possible. If Γ is negative, this kind of problem is not there, because we can always perform PV curtailment and produce less. The other components of the system stays the same.

A Graphs

A.1 60 Prosumers

Table 4: Profiles

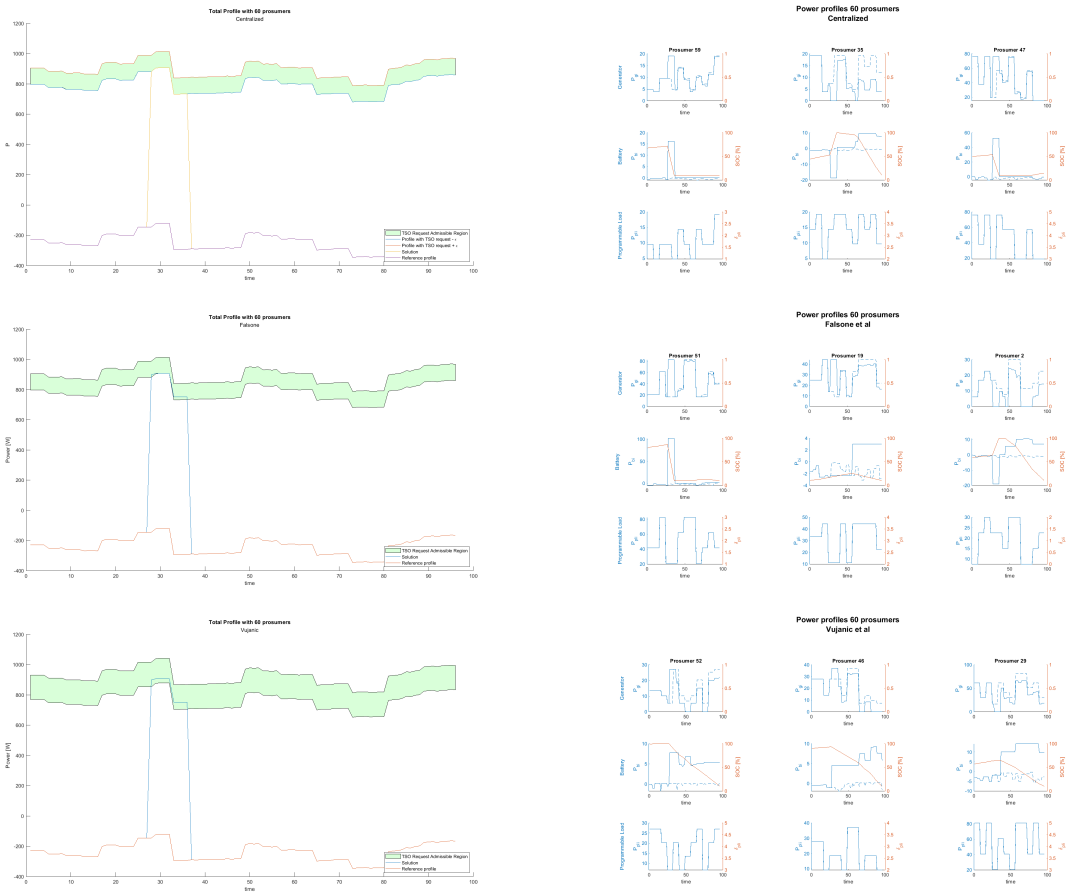
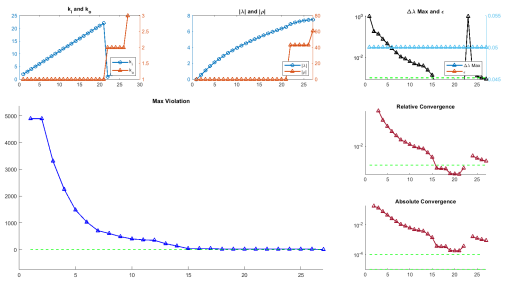
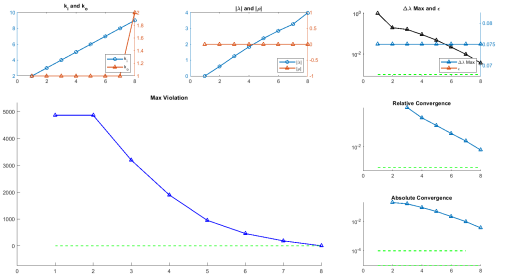


Table 5: Solution progress

Falsone et al.



Vujanic et al



A.2 70 Prosumers

Table 6: Profiles

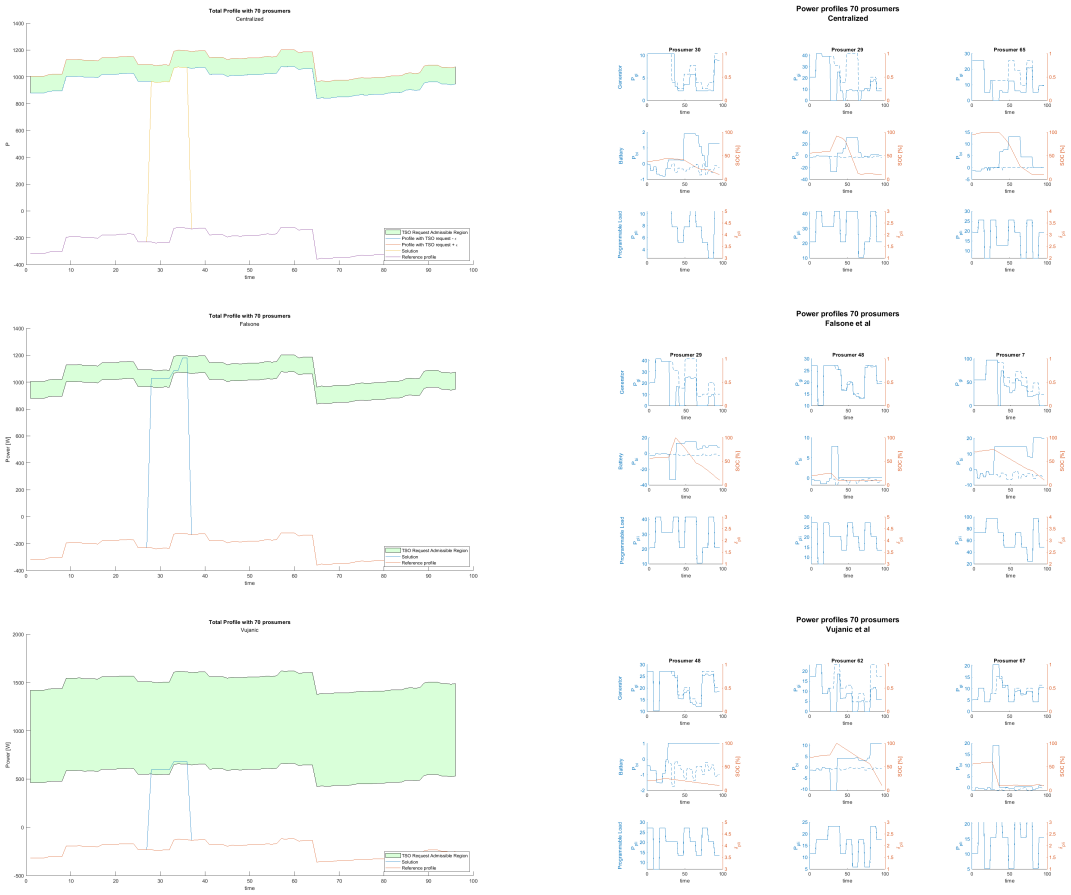
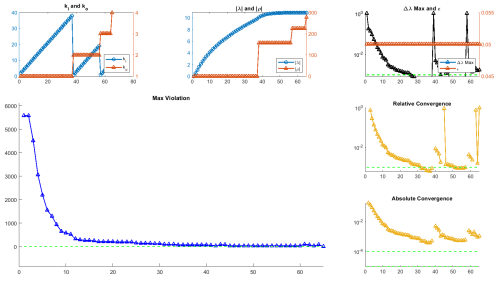
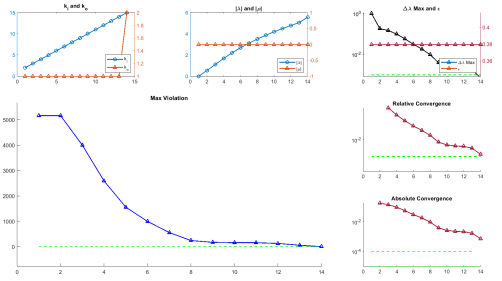


Table 7: Solution progress

Falsone et al.



Vujanic et al



A.3 80 Prosumers

Table 8: Profiles

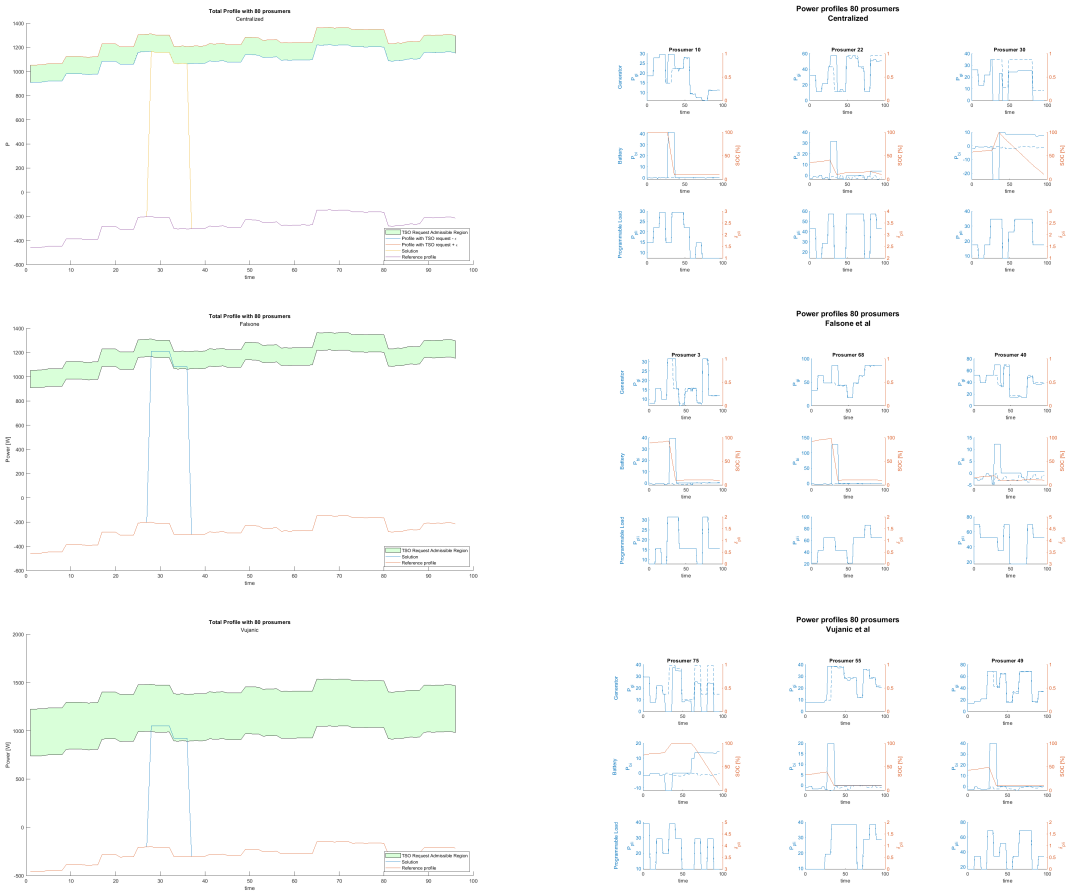
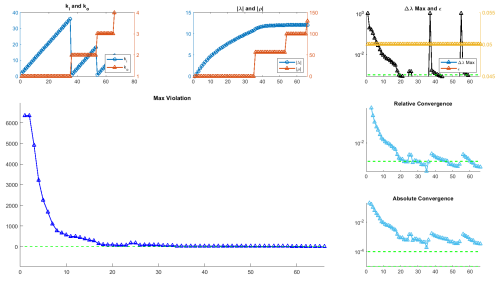
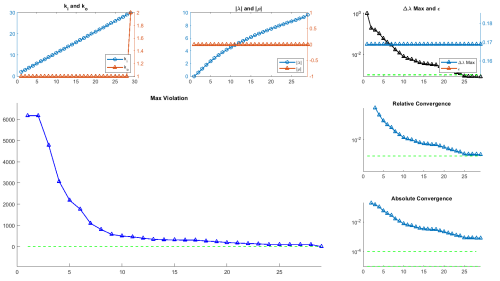


Table 9: Solution progress

Falsone et al.



Vujanic et al



A.4 90 Prosumers

Table 10: Profiles

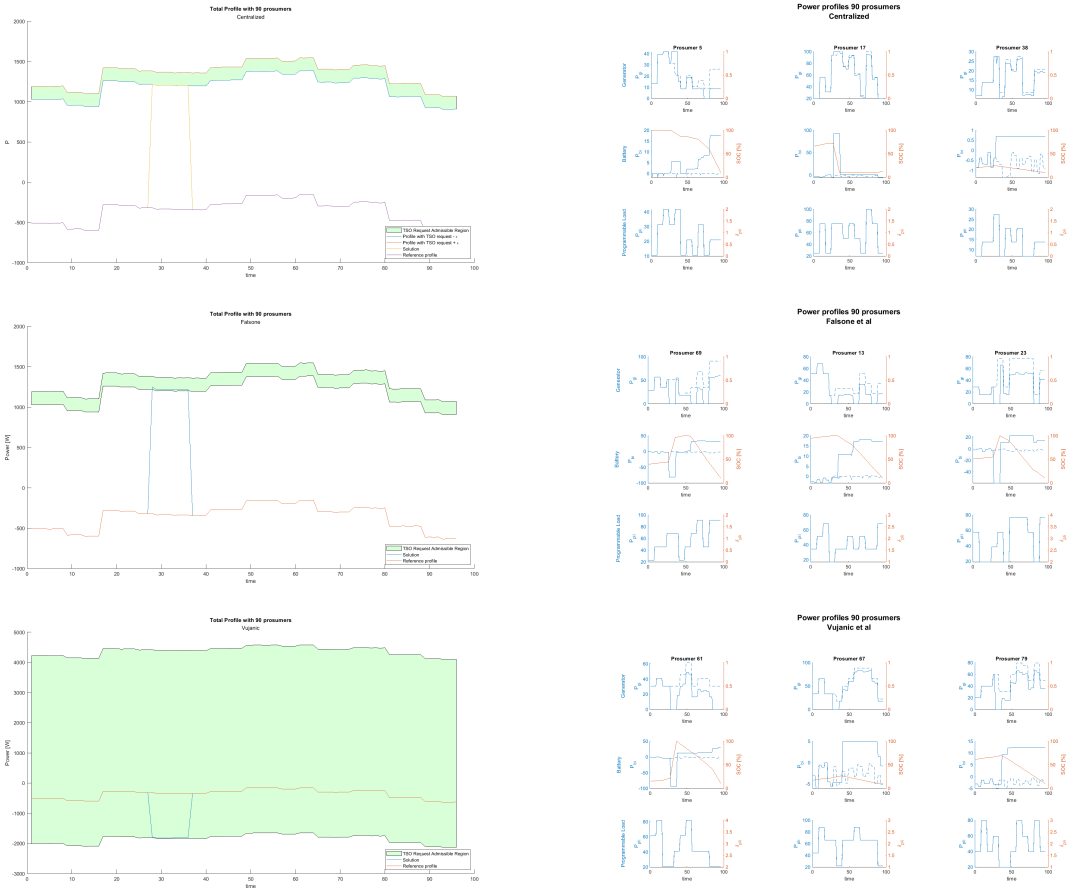
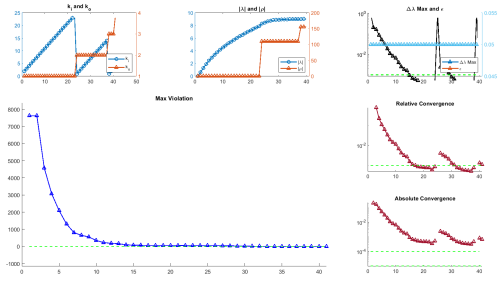
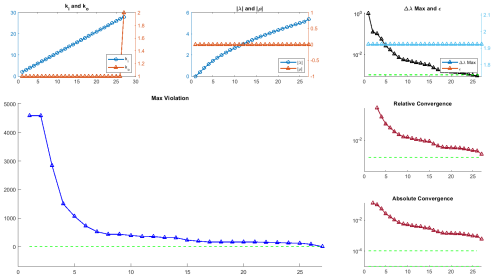


Table 11: Solution progress

Falsone et al.



Vujanic et al



A.5 100 Prosumers

Table 12: Profiles

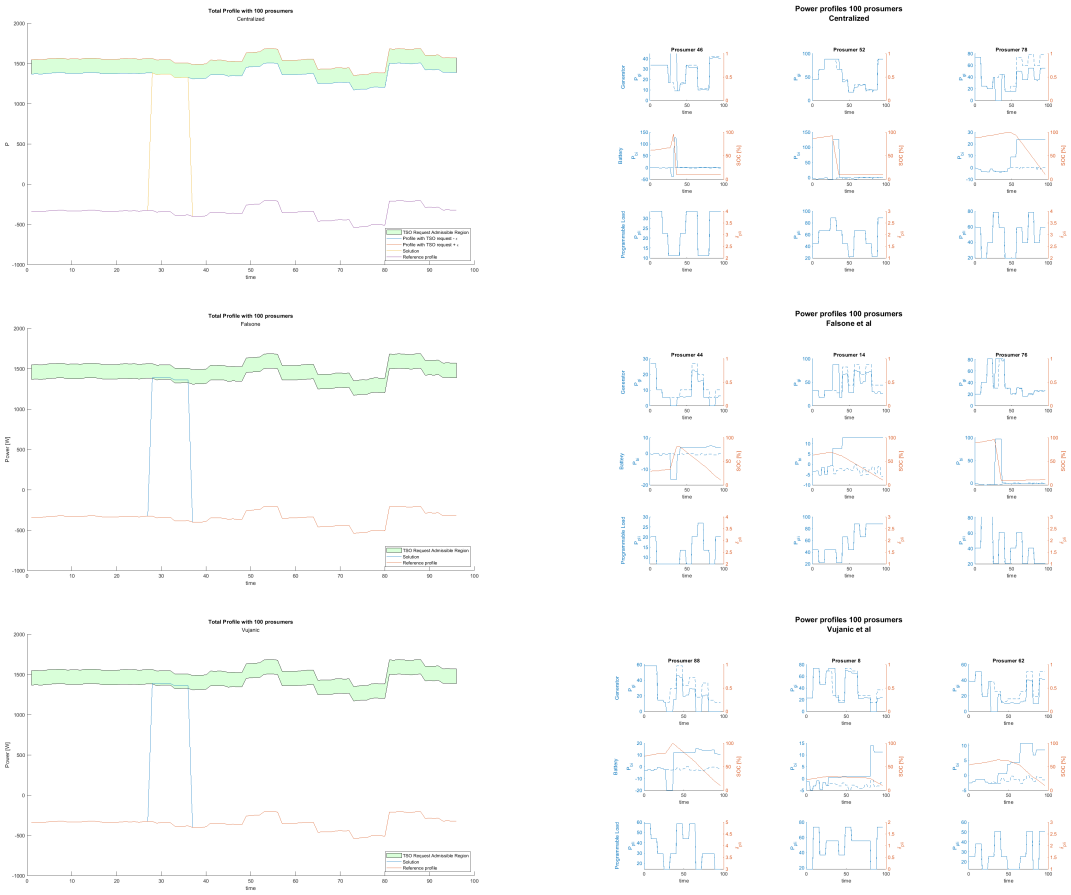
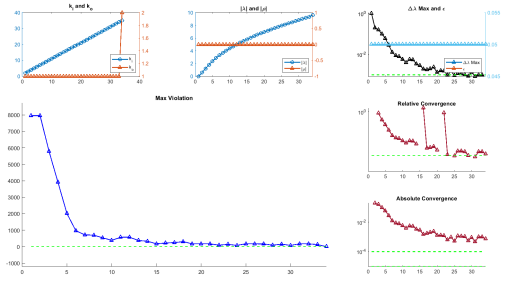
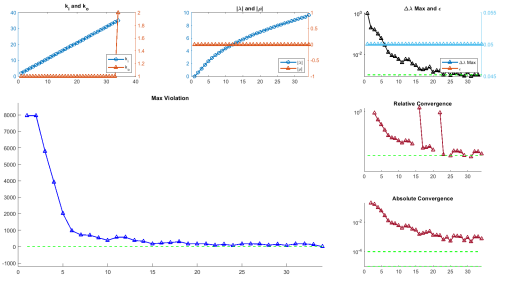


Table 13: Solution progress

Falsone et al.

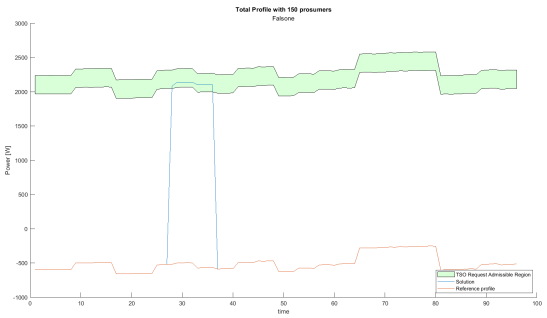


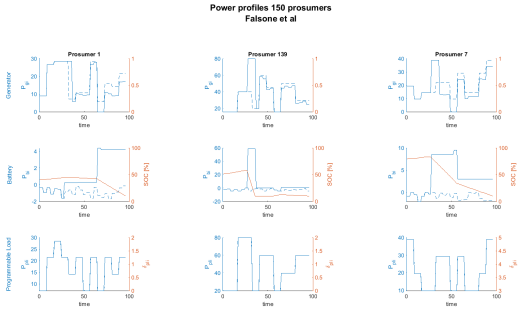
Vujanic et al

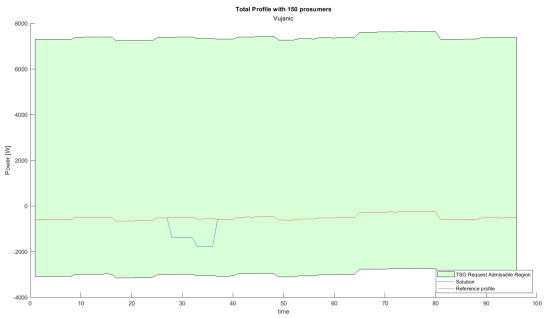


A.6 150 Prosumers

Table 14: Profiles







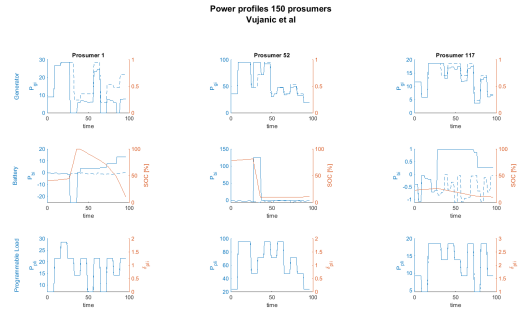
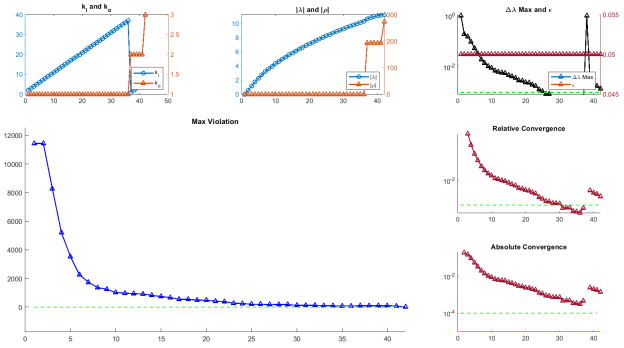
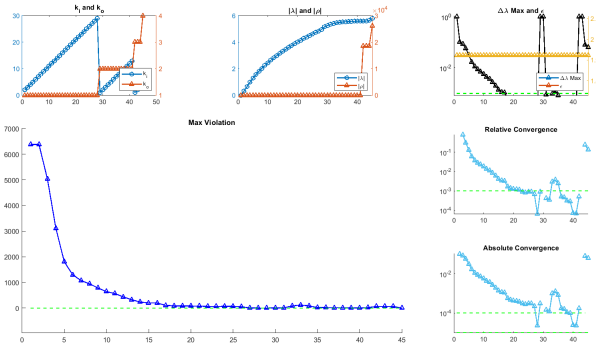


Table 15: Solution progress

Falsone et al.

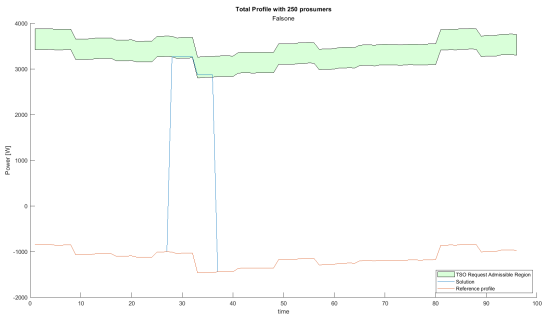


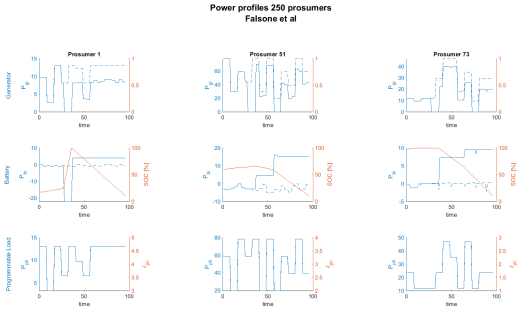
Vujanic et al

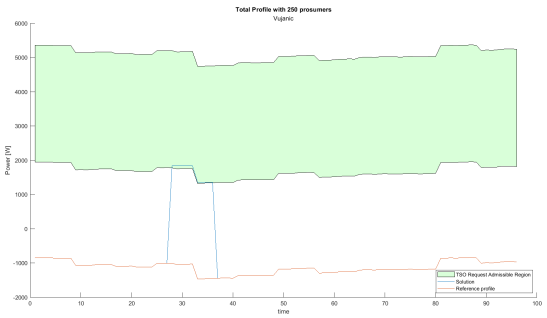


A.7 250 Prosumers

Table 16: Profiles







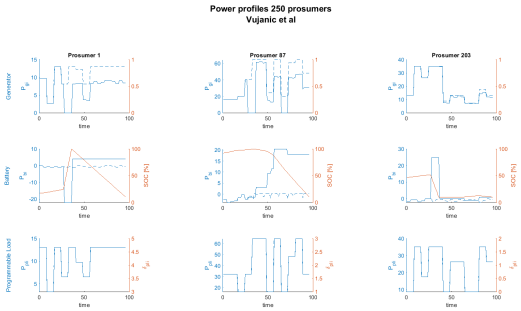
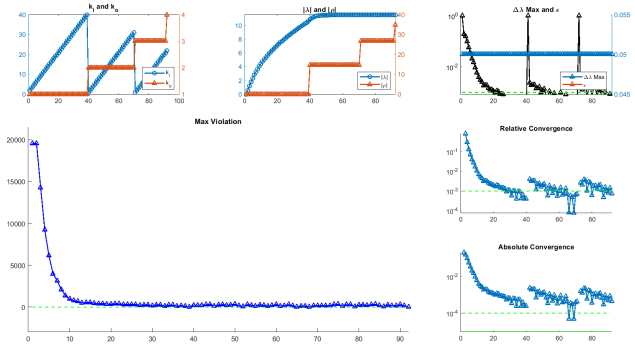
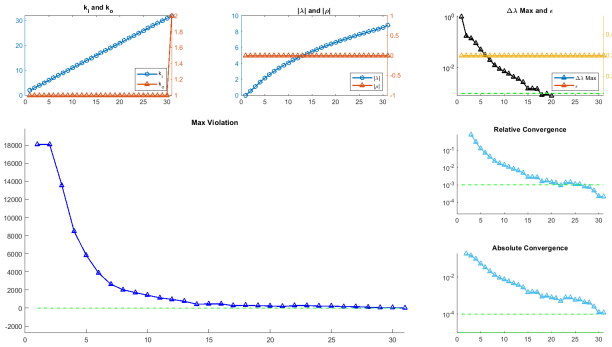


Table 17: Solution progress

Falsone et al.

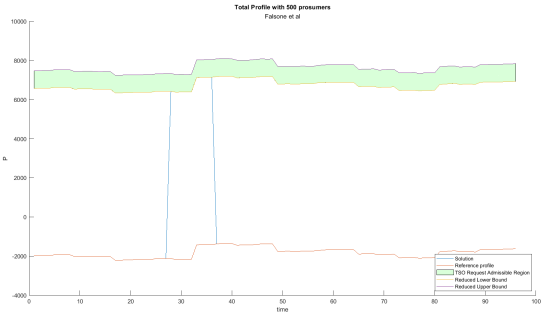


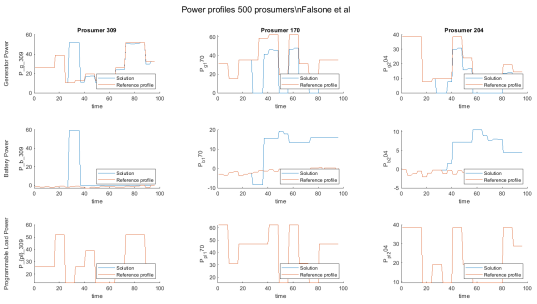
Vujanic et al

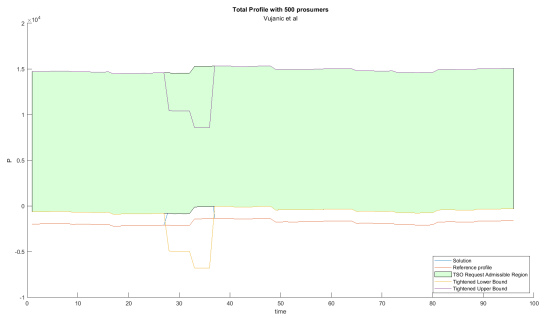


A.8 500 Prosumers

Table 18: Profiles







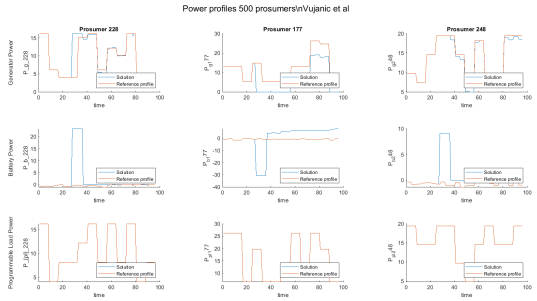
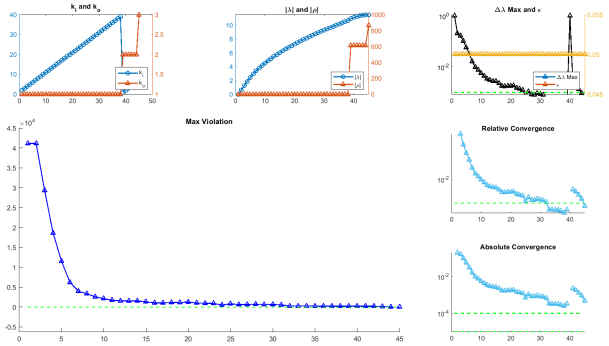
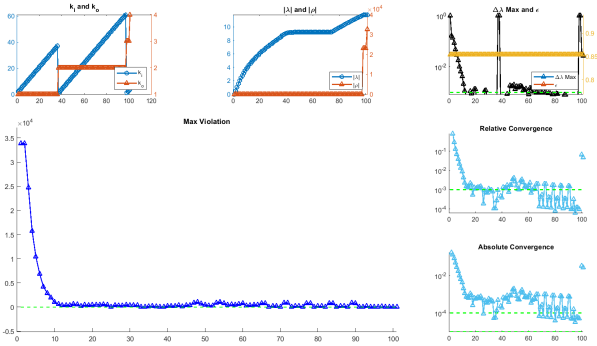


Table 19: Solution progress

Falsone et al.

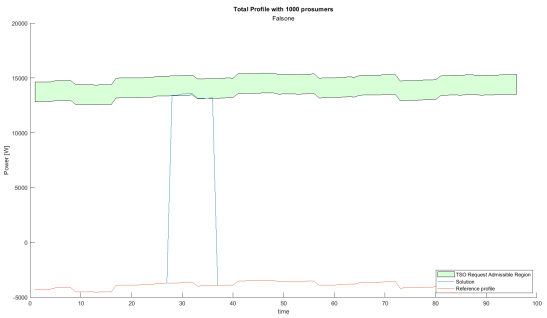


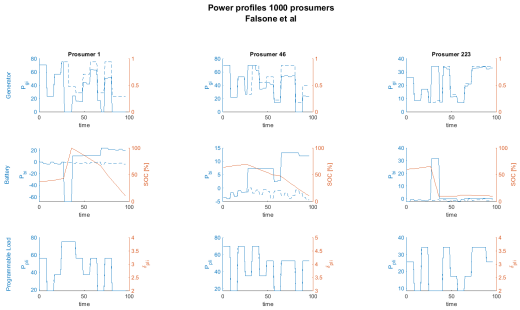
Vujanic et al

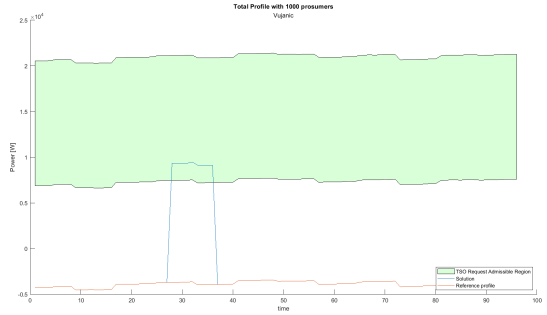


A.9 1000 Prosumers

Table 20: Profiles







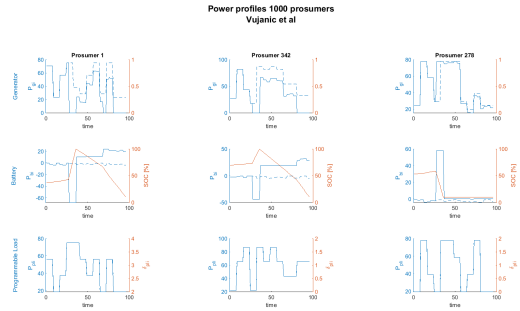
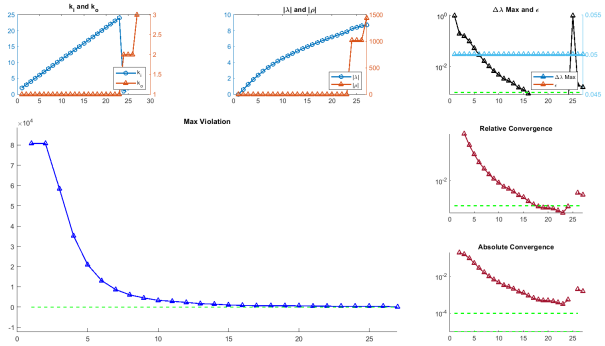
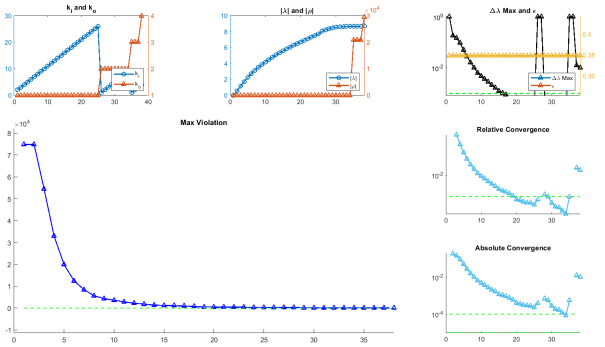


Table 21: Solution progress

Falsone et al.



Vujanic et al



A.10 2000 Prosumers

Table 22: Profiles

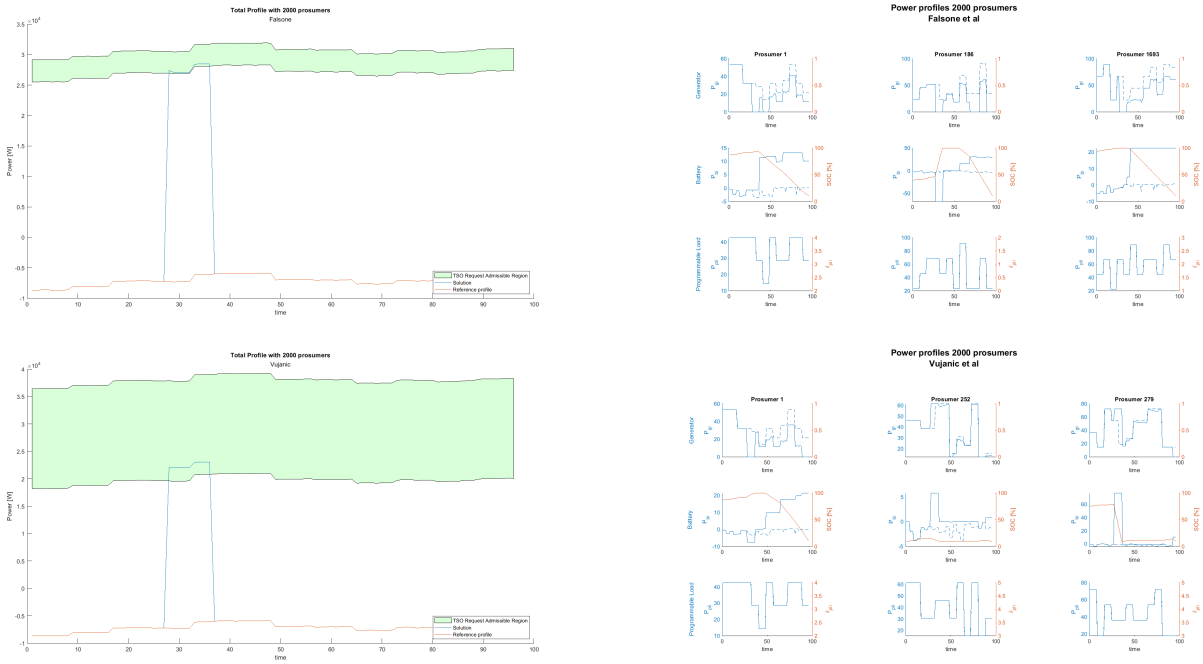
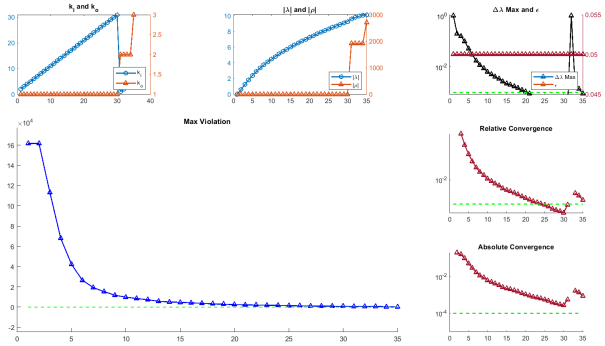
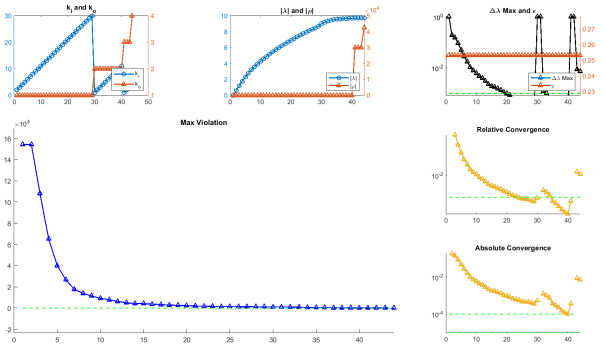


Table 23: Solution progress

Falsone et al.

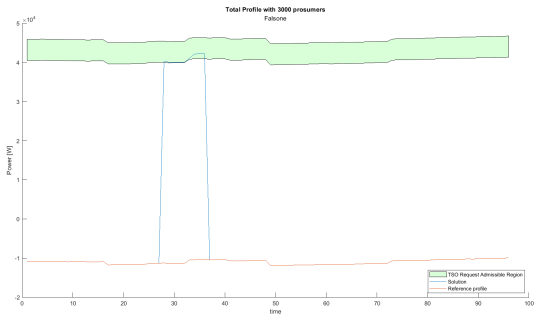


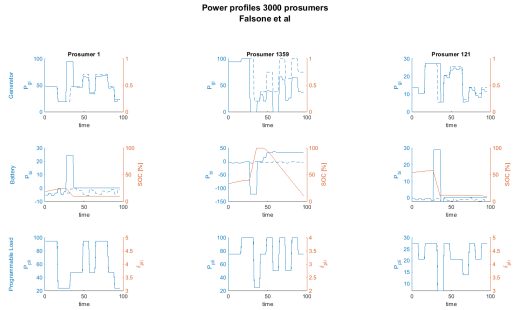
Vujanic et al

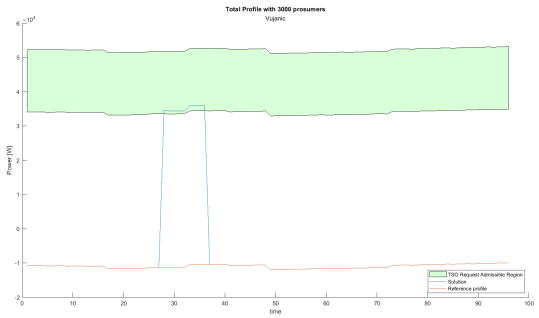


A.11 3000 Prosumers

Table 24: Profiles







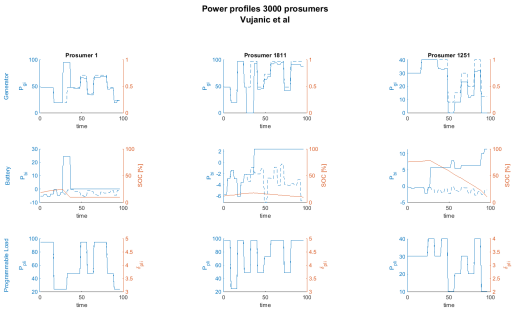
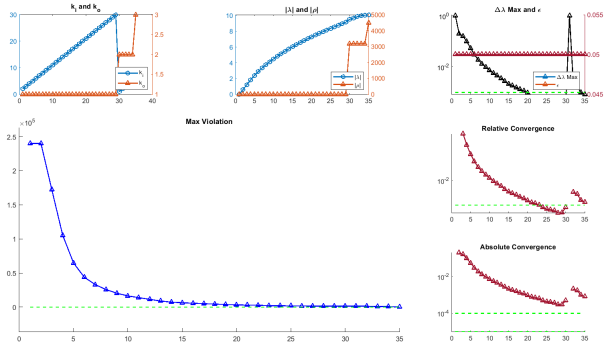
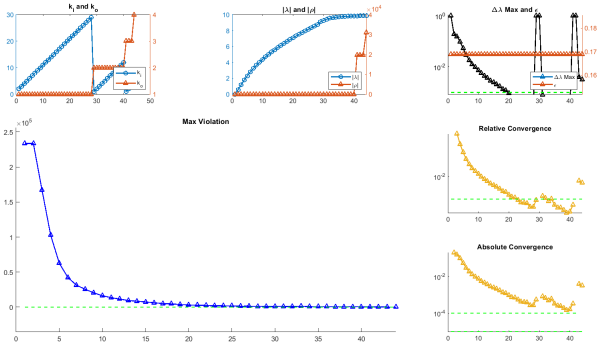


Table 25: Solution progress

Falsone et al.

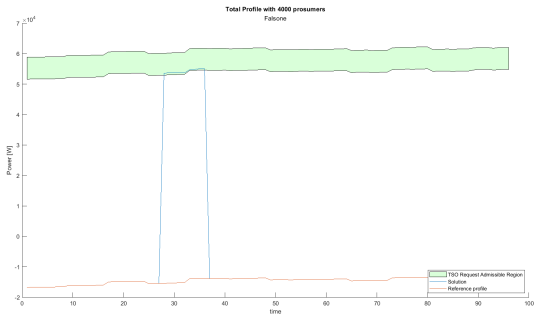


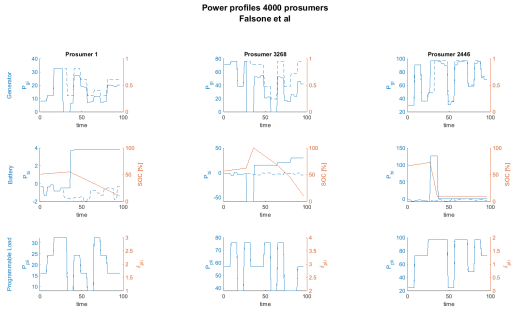
Vujanic et al

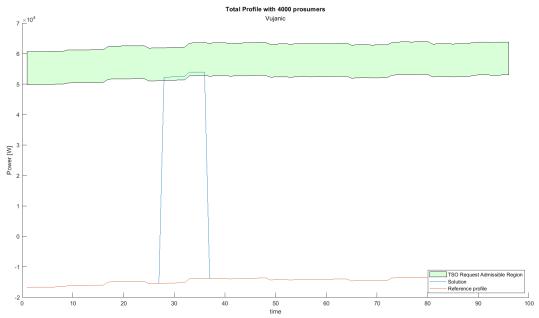


A.12 4000 Prosumers

Table 26: Profiles







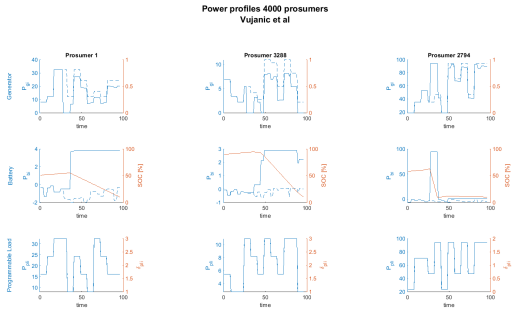
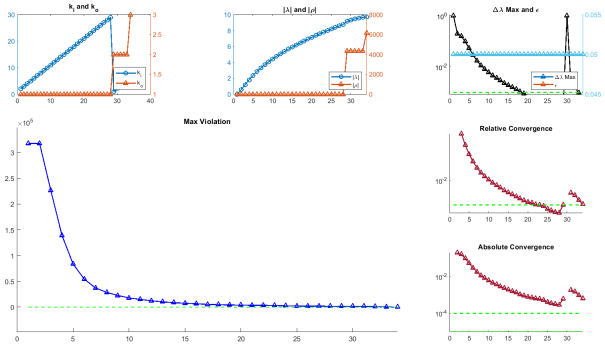
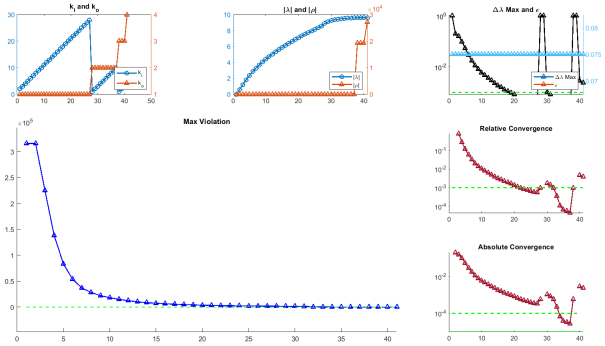


Table 27: Solution progress

Falsone et al.

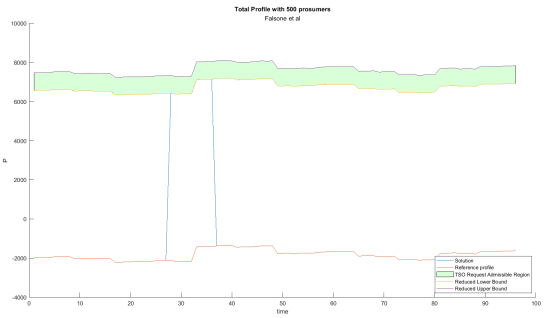


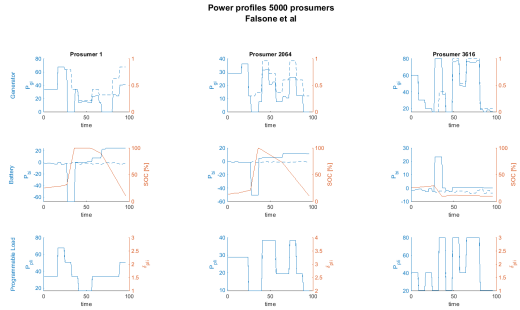
Vujanic et al

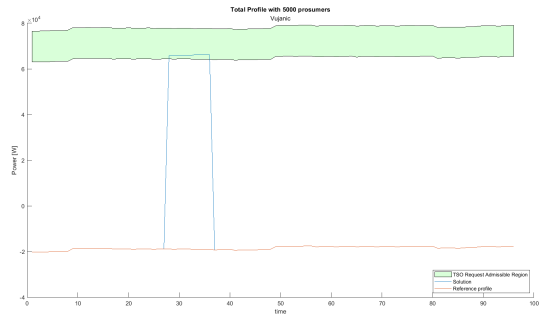


A.13 5000 Prosumers

Table 28: Profiles







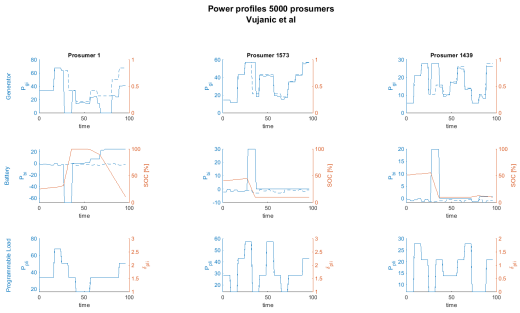
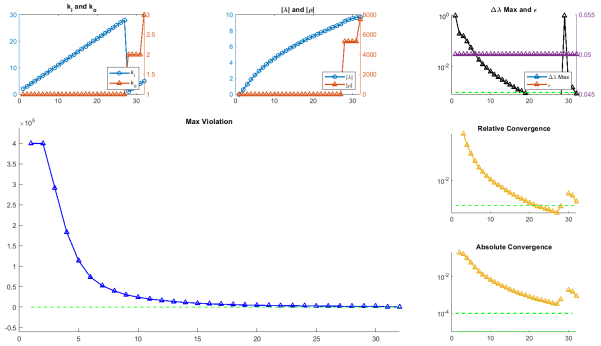
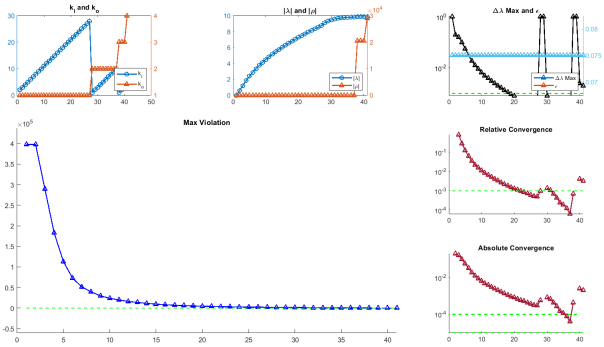


Table 29: Solution progress

Falsone et al.



Vujanic et al



References

Falsone, Alessandro, Kostas Margellos, and Maria Prandini (2019). "A decentralized approach to multi-agent MILPs: Finite-time feasibility and performance guarantees". In: *Automatica* 103, 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.

La Bella, Alessio et al. (2021). "A mixed-integer distributed approach to prosumers aggregation for providing balancing services". In: *International Journal of Electrical Power & Energy Systems* 133, p. 107228. ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2021.107228. URL: https://www.sciencedirect.com/science/article/pii/S0142061521004671.

Manieri, Lucrezia, Alessandro Falsone, and Maria Prandini (Not yet published). "Handling complexity in large scale cyber-physical systems through distributed computation". In.

Vujanic, Robin et al. (2016). "A decomposition method for large scale MILPs, with performance guarantees and a power system application". In: *Automatica* 67, 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.