Decentralized Algorithms for Generalized Nash Equilibrium Seeking in Peer-to-peer Electricity Markets

Giuseppe Belgioioso

October 2, 2020

1 Problem statement

Consider the same problem statement in `problem_setup_v6.tex`.

2 Algorithms

2.1 Semi-decentralized

The algorithm presented in this section corresponds to [**?**, Alg. 6B] applied on `problem_setup_v6.tex`. Consider the following choices for the step-sizes in Algorithm 1.

Assumption 1 (Step-size) *Set the step sizes of prosumers and DSO as follows:*

(i) $\forall i \in \mathcal{N}$: *set* $A_i = \mathrm{diag}(\alpha_i^{pi}, \alpha_i^{st}, \alpha_i^{mg}, \{\alpha_{(i,j)}^{tr}\}_{j \in \mathcal{N}_i}) \otimes I_H$, *with* $\alpha_i^{pi}, \alpha_i^{st} > 1$, $\alpha_i^{mg} > 3 + N \max_{h \in \mathcal{H}} d_h^{mg}$, $\alpha_{(i,j)}^{tr} > 2$, $\forall j \in \mathcal{N}_i$, $\beta_{(i,j)}^{tr} = \beta_{(j,i)}^{tr} < \frac{1}{2}$, $\forall j \in \mathcal{N}_i$.

(ii) *Set* $A_{N+1} := \mathrm{diag}\left(\left\{\alpha_y^{\theta}, \alpha_y^{v}, \alpha_y^{tg}, \{\alpha_{(y,z)}^{p}, \alpha_{(y,z)}^{q}\}_{z \in \mathcal{B}_y}\right\}_{y \in \mathcal{B}}\right) \otimes I_H$, *with* $\alpha_y^{\theta}, \alpha_y^{v} > 0$, $\alpha_y^{tg} > 2$, $\alpha_{(y,z)}^{p} > 1$ *and* $\alpha_{(y,z)}^{q} > 0$, $\forall z \in \mathcal{B}_y$, $\forall y \in \mathcal{B}$. *Set* $\gamma^{mg} < \frac{1}{N}$, $\beta^{tg} < (|\mathcal{N}| + |\mathcal{B}|)^{-1}$ *and* $\beta_y^{pb} < (1 + |\mathcal{N}_y| + |\mathcal{B}_y|)^{-1}$, *for all* $y \in \mathcal{B}$. □

Algorithm 1 Semi-decentralized GWE seeking for P2P Energy Markets

---

1: **Iterate until convergence**
2:    **for all** prosumer $i \in \mathcal{N}$ **do**
3:       **primal update**   ▷ power generated, stored, from the grid, traded
4:           $a_i(k) = \mathrm{col}\left(-\mu_y^{\mathrm{pb}}(k), -\mu_y^{\mathrm{pb}}(k), \begin{bmatrix} I_H \\ -I_H \end{bmatrix}^{\top} \lambda^{\mathrm{mg}}(k) + \mu^{\mathrm{tg}}(k), \left\{\mu_{(i,j)}^{\mathrm{tr}}(k)\right\}_{j \in \mathcal{N}_i}\right)$ ▷ aux. vector
5:           $u_i(k+1) = \begin{cases} \underset{\xi \in \mathbb{R}^{n_i}}{\mathrm{argmin}} & J_i\big(\xi, \sigma(k)\big) + a_i(k)^{\top}\xi + \frac{1}{2}\|\xi - u_i(k)\|_{A_i}^2 \\ \text{s.t.} & \xi \in \mathcal{U}_i \end{cases}$ ▷ quadratic progr.
6:       **end**
7:       **communication**   ▷ to DSO and trading partners
8:           $b_i(k+1) = p_i^{\mathrm{d}} - p_i^{\mathrm{di}}(k+1) - p_i^{\mathrm{st}}(k+1)$ ▷ local load unbalance of prosumer i
9:           $p_i^{\mathrm{mg}}(k+1), b_i(k+1) \longrightarrow \mathrm{DSO}$,   ▷ forward to DSO
10:           **for all** prosumer $j \in \mathcal{N}_i$ **do**
11:              $p_{(i,j)}^{\mathrm{tr}}(k+1) \longrightarrow$ prosumer j ▷ forward local trade to prosumer j
12:           **end for**
13:       **end**
14:       **dual update**   ▷ reciprocity constraints
15:           **for all** $j \in \mathcal{N}_i$ **do**
16:              $c_{(i,j)}^{\mathrm{tr}}(k+1) = p_{(i,j)}^{\mathrm{tr}}(k+1) + p_{(j,i)}^{\mathrm{tr}}x(k+1)$ ▷ aux. vector
17:              $\mu_{(i,j)}(k+1) = \mu_{(i,j)}(k) + \beta_{ij}^{\mathrm{tr}}\left(2c_{(i,j)}^{\mathrm{tr}}(k+1) - c_{(i,j)}^{\mathrm{tr}}(k)\right)$ ▷ reflected dual ascent
18:           **end for**
19:       **end**
20:    **end for**
21:    **DSO update**
22:       **primal update**   ▷ physical variables
23:           $a_{N+1}(k+1) = \mathrm{col}\left(\left\{\mathbf{0}, \mathbf{0}, -\mu^{\mathrm{tg}}(k), \{\mu_y^{\mathrm{pb}}(k), \mathbf{0}\}_{z \in \mathcal{B}_y}\right\}_{y \in \mathcal{B}}\right)$ ▷ aux. vector
24:           $u_{N+1}(k+1) = \mathrm{proj}_{\mathcal{U}_{N+1}}\left(u_{N+1}(k) - A_{N+1}a_{N+1}(k)\right)$ ▷ solved via Algorithm 2
25:       **end**
26:       **aggregation update**
27:           $\sigma^{\mathrm{mg}}(k+1) = \sum_{i \in \mathcal{N}} p_i^{\mathrm{mg}}(k+1)$ ▷ aggregate grid-to-prosumers power
28:           $\sigma^{\mathrm{tg}}(k+1) = \sum_{y \in \mathcal{B}} p_y^{\mathrm{tg}}(k+1)$ ▷ aggregate grid-to-buses power
29:       **end**
30:       **dual update**
31:           $b_{N+1}(k+1) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes (2\sigma^{\mathrm{mg}}(k+1) - \sigma^{\mathrm{mg}}(k)) - \begin{bmatrix} \overline{p}^{\mathrm{mg}}\mathbf{1}_H \\ -\underline{p}^{\mathrm{mg}}\mathbf{1}_H \end{bmatrix}$ ▷ aux. vector
32:           $\lambda^{\mathrm{mg}}(k+1) = \mathrm{proj}_{\mathbb{R}_{\geq 0}^{2H}}\left(\lambda^{\mathrm{mg}}(k) + \gamma^{\mathrm{mg}}b_{N+1}(k+1)\right)$ ▷ grid constraints
33:           **for all** buses $y \in \mathcal{B}$ **do**
34:              $c_y^{\mathrm{pb}}(k+1) = p_y^{\mathrm{pd}} + \sum_{i \in \mathcal{N}_y} b_i(k+1) - p_y^{\mathrm{tg}}(k+1) - \sum_{z \in \mathcal{B}_y} p_{(y,z)}^{\ell}(k+1)$ ▷ aux. vector
35:              $\mu_y^{\mathrm{pb}}(k+1) = \mu_y^{\mathrm{pb}}(k) + \beta_y^{\mathrm{pb}}(2c_y^{\mathrm{pb}}(k+1) - c_y^{\mathrm{pb}}(k))$ ▷ local power balance of bus y
36:           **end for**
37:           $c^{\mathrm{tg}}(k+1) = \sigma^{\mathrm{mg}}(k+1) - \sigma^{\mathrm{tg}}(k+1)$ ▷ aux. vector
38:            $\mu^{\mathrm{tg}}(k+1) = \mu^{\mathrm{tg}}(k) + \beta^{\mathrm{tg}}(2c^{\mathrm{tg}}(k+1) - c^{\mathrm{tg}}(k))$ ▷ grid-to-buses constraints
39:       **end**
40:       **communication**   ▷ broadcast
41:           $\{\sigma(k+1), \lambda^{\mathrm{mg}}(k+1), \mu^{\mathrm{tg}}(k+1)\} \longrightarrow \mathcal{N}$ ▷ to all prosumers
42:           **for all** buses $y \in \mathcal{B}$ **do**
43:              $\mu_y^{\mathrm{pb}}(k+1) \longrightarrow \mathcal{N}_y$ ▷ to all prosumers on bus y
44:           **end for**
45:       **end**
46:    **end**
47: **end**

2.2 Projection onto $\mathcal{U}_{N+1}$

Next, we propose an iterative method to solve line 24 in Algorithm 1, namely, to compute the projection onto $\mathcal{U}_{N+1}$. First, let us define the sets $C_1 := (17) \cap (18) \cap (19) \cap (20)$ and $C_2 = (15) \cap (16)$ and recall that the decision vector of the DSO reads as $u_{N+1} = \mathrm{col}\left(\{\theta_y, v_y, p_y^{\mathrm{tg}}, \{p_{(y,z)}^{\ell}, q_{(y,z)}^{\ell}\}_{z \in \mathcal{B}_y}\}_{y \in \mathcal{B}}\right)$. The projection onto C_1 can be characterize in closed-form as follows:

$$\mathrm{proj}_{C_1}(u_{N+1}) = \mathrm{col}\left(\{\theta_y^+, v_y^+, p_y^{\mathrm{tg}+}, \{{p_{(y,z)}^{\ell}}^+, {q_{(y,z)}^{\ell}}^+\}_{z \in \mathcal{B}_y}\}_{y \in \mathcal{B}}\right),$$

where, for all $y \in \mathcal{B}$,

$$\theta_y^+ = \begin{cases} \underline{\theta}_y, & \text{if } \theta_y < \underline{\theta}_y \\ \overline{\theta}_y, & \text{if } \theta_y > \overline{\theta}_y \\ \theta_y, & \text{otherwise} \end{cases},$$

$$v_y^+ = \begin{cases} \underline{v}_y, & \text{if } v_y < \underline{v}_y \\ \overline{v}_y, & \text{if } v_y > \overline{v}_y \\ v_y, & \text{otherwise} \end{cases}$$

$$p_y^{\mathrm{tg}+} = \begin{cases} p_y^{\mathrm{tg}}, & \text{if } y \in \mathcal{B}^{\mathrm{mg}} \\ 0, & \text{otherwise} \end{cases}$$

$$(p_{(y,z),h}^{\ell})^+ = \frac{\overline{s}_{(y,z)}}{\max\left\{\|\mathrm{col}(p_{(y,z),h}^{\ell}, q_{(y,z),h}^{\ell})\|, \overline{s}_{(y,z)}\right\}} p_{(y,z),h}^{\ell}, \qquad \forall z \in \mathcal{B}_y, \forall h \in \mathcal{H}$$

$$(q_{(y,z),h}^{\ell})^+ = \frac{\overline{s}_{(y,z)}}{\max\left\{\|\mathrm{col}(p_{(y,z),h}^{\ell}, q_{(y,z),h}^{\ell})\|, \overline{s}_{(y,z)}\right\}} q_{(y,z),h}^{\ell}, \qquad \forall z \in \mathcal{B}_y, \forall h \in \mathcal{H}$$

The projection onto C_2 can be computed by solving a quadratic programming (e.g. via lsqlin, quadprog, osqp, etc...) with appropriate matrices.

Algorithm 2 Douglas–Rachford splitting to compute the projection of x onto $\mathcal{U}_{N+1} = C_1 \cap C_2$

1: **Iterate until convergence**
2:      $z(k) = \mathrm{proj}_{C_1}(\frac{1}{2}\xi(k) + \frac{1}{2}x)$
3:      $\xi(k+1) = \xi(k) + \lambda\left(\mathrm{proj}_{C_2}(2z(k) - \xi(k)) - z(k)\right),$ with $\lambda \in (0,2)$
4: **end**