Distributed Optimization Final Project

Andrew Wentzel

1 Introduction

This paper deals with some of the algorithms for optimal resource pricing in theoretical smart-grid systems. While a number of algorithms and paradigms have been presented, he we will deal with a specific subset dealing with demand-side management in the idealized case of a steady-state system. The Structure of the paper is as follows:

- Section I Introduces the motivation and mathematical model of the problem formulation.
- Section II Introduces and discusses 3 algorithms proposed in literature. Modifications are introduced and discussed as well.
- Section III Deals with implementation of the 3 discussed algorithms in Python on a model system and discusses their performance.
- Section IV Summarized the results as well as discusses other related problems and algorithms relevant to smart-grids.

1.1 Motivation

Advancements in technology and sustainability has opened up opportunities for new paradigms in energy management. Modern energy systems that traditionally rely on utilities that have a regulated monopoly on energy production have moved towards and increasingly decentralized system relying on local data collection and smaller, independent energy producers. The future of the energy-grid is envisioned to be a dynamic and distributed system. Such advances have opened up room to explore problems relating to optimal control of smart-grids.

One such problem is known as the "Economic Dispatch Problem" (EDP). In this problem, the smart-grid is modeled as a graph composed of nodes. Each node has a local resource demand, with a subset of these nodes having a set resource production capacity, which will be referred to as generators. For these nodes the cost of producing the resource is usually modeled as a quadratic cost function. This problem deals with the question of the optimal "pricing" of the resource - finding a universal price to "pay" each generator so as to minimize the overall cost while meeting the total demand over the whole grid. To more concretely present the problem formulation, we will first introduce some basic notation:

$$G_m = (V_m, E_m) \quad \text{is a strongly-connected graph of all m nodes}$$

$$G_n = (V_n, E_n) \quad \text{is a strongly-connected graph of all n generators}$$

$$s.t.$$

$$V_n \subseteq V_m$$

$$x_i \quad \text{is the amount of the resource produced at generator i}$$

$$P_i \quad \text{is the constant resource demand at node i}$$

$$P^* \quad \text{is the total demand over the whole graph}$$

$$f_i(x_i) \quad \text{is the cost of producing a given amount at at generator i}$$

The problem can now be written as:

$$min \sum_{i \in V_n} f_i(x_i)$$

$$s.t. \quad \sum_{i \in V_m} x_i = P^*$$

With the additional constraint that x_i can be bounded above and below:

$$\underline{\mathrm{x}}_i < x_i < \bar{x}^i$$

$f_i(x)$ is usually modelled as a quadratic cost function:

$$f_i(x) = a_i x^2 + b_i x + c$$

Which is commonly re-written as:

$$f_i(x) = \frac{x - \alpha_i}{\beta_i}$$

1.2 Centralized Solution

The centralized solution to the EDP is well known. To simplify the solution we will consider first the unconstrained case, and note that the constrained solution can be found by simple projection into the feasible space of x. The dual problem can now be written as:

$$max \sum_{i \in V_n} f_i(x_i) + \lambda\big(P^* - x_i\big)$$

From the kkt criteria the dual solution can be found. Let:

$$u_i = \frac{df_i(x_i)}{dx_i}$$

$$\lambda^* = u_i(x_i^*)$$

Here u_i is known as the incremental cost. From the definition of the cost function, we have that the problem formulation is strictly convex and continuous, along with affine constraints. If we assume that a feasible solution exists, we can ensure strong duality from Slater's condition. Thus, the solution in the primal space is written as:

$$x_i^* = u_i^{-1}(\lambda^*)$$

For solving the solution in the distributed case, the solution method can be easily extended to account for bounds on x. Since we will be iterative solving for λ and x, we can define a mapping of x into the feasible space. That is:

$$C(x) = \begin{cases} \underline{\mathrm{x}}, & \text{if } x < \underline{\mathrm{x}}. \\ \bar{x}, & \text{if } x > \underline{\mathrm{x}}. \\ x, & \text{otherwise.} \end{cases} \tag{1}$$

Thus, we can solve the problem in the unconstrained case, map $x^* \to C(x^*)$ and then solve for the optimal Lagrange multiplier as $\lambda^* = u(C(x))$. The projection operations $C(x)$ preserves the monotonic behavior of the function, and thus allows for the convergence criteria for the following algorithm to hold.

It is worth noting that while the optimal solution to the problem is written in terms of x, the control variable we are trying to optimize is λ. Here λ is representative of the pricing of the given resource. Thus, if we assume that each individual generator seeks to maximize profit, the resulting power generator at each node when the pricing is λ will be x_i^*. The feasible optimal solution then gives us a pricing structure so that all demand will be met while minimizing total cost.

2 Algorithms

This section introduces 3 algorithms for solving the EDP in the distributed case. These algorithms seek to solve the problem in the case that each node is only aware of local information. That is, it can share information with the nodes directly adjacent to each other, without knowledge of the overall grid or the individual cost functions of the other nodes. In order to solve this, each generator will need some estimate of the total load, which we will call the virtual demand at each node. Many algorithms rely on the idea of a single leader node that is aware of the total demand. The algorithms in this paper are specifically ones where it is sufficient that each generator have an estimate y_i, called the *virtual demand* such that:

$$\sum_{i \in V_n} y_i = P^*$$

To do this, we will first introduce a consensus algorithm for this introduced in [5] that maps the total demand at each node into virtual demand at each generator. The benefit of this is that the computational efficiency of the main algorithm goes down significantly, as much fewer nodes are needed for the computation, at the cost of computational overhead in the transformation.

2.1 Algorithm 0: Virtual Demand Algorithm

To begin we need to define two normalized Adjacency Matrices, Q and R such that:

$$r_{i,j} = \begin{cases} \frac{1}{d^+_{n,j}+1}, & \text{if } (i,j) \in E_n. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

$$q_{i,j} = \begin{cases} \frac{1}{d^+_{m,j}+1}, & \text{if } (i,j) \in E_m. \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

It should be noted that these graphs are column stochastic. Thus, running the power-method style iteration on a vector v with the Matrix $M \in [Q, R]$ will yield:

$$Mv^* = v^*$$

$$\|v^*\| = \|v\|$$

Recall that P_i is the initial demand among the entire grid. We will then initialize $p_i^0 = P_i$ and run the iteration:

$$p_i^{t+1} = q_{ii} p_i^t + \sum_{i \in V_m} q_{ij} p_j^t$$

Once p converges, a new variable s is introduced:

$$s_i^0 = \begin{cases} p_i^*, & \text{if } i \in E_n. \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

And We run:

$$s_i^{t+1} = q_{ii} s_i^t + \sum_{i \in V_m} q_{ij} s_j^t$$

This converges to a scaling variable that we can use as a scaling variable between graph G_m and G_n. We do this by initializing our final variable:

$$y_i^0 = \frac{(p_i^*)^2}{s_i^*}, \quad i \in V_n$$

It should be noted that y_i is only defined in the space of G_n. That is, all generator nodes. We then run the update:

$$y_i^{t+1} = r_{ii} y_i^t + \sum_{i \in V_n} r_{ij} y_j^t$$

This will converge to

$$y_i^* = P^* v_i$$

Where $v = [v_1, v_2, ..., v_n]$ is the right, normalized eigenvector of R corresponding to the eigenvalue of 1 and a magnitude of 1.

Assuming that each node in our graph has an estimate of virtual demand y_i, we can continue on to the main algorithms.

2.2 Algorithm 1 - Bisection

This algorithm is introduced in [5]. We start with the assumption that the solution λ^* is feasible. An important idea of the problem formulation is that the cost function $f_i(x)$ is twice differentiable positive such that:

$$\frac{d^2 f_i(x)}{dx^2} > 0$$

Since $\lambda = u_i(x)$, we get that λ will be monotonically increasing over set of feasible x. Thus, the feasible set of possible values for λ is bounded below by the minimum of feasible incremental costs across all generators, and bounded above by the corresponding maximum incremental cost. Concretely, we define these initial upper and lower bounds as:

$$\lambda_0^- = min\, u_i(\underline{x}_i)$$

$$\lambda_0^+ = max\, u_i(\bar{x}_i)$$

This initialization requires that all generator nodes come to a consensus on these feasible bounds. While the paper does not discuss this directly, this problem can be solved withing D steps, where D is the maximum distance between any two nodes on the graph with:

$$\lambda_i^- = u_i(\underline{x}_i)$$

$$\lambda_i^+ = u_i(\bar{x}_i)$$

With The update:

$$\lambda_{i+1}^- = min\, u_j(\underline{x}_j), \quad j \in N_i^-$$

$$\lambda_{i+1}^+ = max\, u_j(\bar{x}_j), \quad j \in N_i^-$$

The midpoint value that bisects the feasible space is our initial guess at λ:

$$\lambda_0 = \frac{\lambda_0^- + \lambda_0^+}{2}$$

The corresponding update law for x is:

$$x_i^k = C\Big(u_i^{-1}(\lambda_k)\Big)$$

For the resulting x, the algorithm then seeks to identify cooperatively if the total resulting supply is more or less than the total demand. For each update, define a variable z such that:

$$z_i^0 = x_i^k$$

Then run the update:

$$z_i^{t+1} = z_{ii} y_i^t + \sum_{i \in V_n} r_{ij} z_j^t$$

As before, this will converge to a value corresponding to:

$$z_i^* = \Big(\sum_{j \in V_n} x_j^k\Big) v$$

Where v is the normalized eigenvector of R with eigenvalue 1. We can see that $z_i^* - y_i$ at each step will correspond to a scaled version of the difference between total supply and demand. The bisection step relies on this and defines the point where:

$$sign(z_i^* - y_i) = sign(z_j^* - y_j) \quad (i,j) \in V_n$$

The update for lambda then becomes:

$$\lambda_k^- = \begin{cases} \lambda_k, & sign(z-y) > 0. \\ \lambda_k^-, & sign(z-y) < 0. \end{cases} \tag{5}$$

$$\lambda_k^+ = \begin{cases} \lambda_k, & sign(z-y) < 0. \\ \lambda_k^+, & sign(z-y) > 0. \end{cases} \tag{6}$$

This bisection can be run until a desired bound on the error is reached. The resulting x is given by:

$$x_i^* = u_i^{-1}(\lambda_k^*)$$

A benefit of this algorithm is that it doesn't require any meta-parameter tuning. Instead, it relies on intermediate steps that ensure global agreement on the proper initial guess and rate of convergence in the graph.

It should also be noted that the special case of fixed-output generators, such as a solar panel, can be incorporating them into the graph of G_m with a fixed negative demand, without a loss of generality. To analyze the convergence rate of the Bisection algorithm we can break it down into its composite parts:

1. The determination of the initial bounds λ_0 is bounded above by the diameter of the graph
2. The step for determining $sign(z-y)$ is bounded below by the diameter of the graph and converges in finite time
3. The update for λ_k converges asymptotically to the optimal value

2.3 Algorithm 2: Consensus + Innovation

The algorithm introduced in [2] approaches the problem in a different way. To begin, we will start our assumptions as before where each generator has access to a variable y. For this algorithm, the process may be run over any set of nodes. It is sufficient here that for the given graph we are computing over, each node has some p such that: $\sum_{i \in V} p_i = P{*}^*$ If we were to consider the case where we are computing over the entire graph, each non-generating node is modeled as a generator bounded above and below by the same value which we will denote by x_{const}, which is 0 in the case of a non-generating node. i.e.

$$\underline{x}_i = \bar{x}_i = x_{const} \quad i \in (V_m \setminus V_n)$$

So the update for x_i at a given λ becomes:

$$x_i = C\Big(u_i^{-1}(\lambda)\Big) = x_{const}$$

For the remainder of this section we will simply assume the algorithm in (2.1) is run, and we are computing over V_n with virtual demands y_i

In The paper the algorithm allows for λ_i^0 be any initial guess. A naive initialization is given by:

$$\lambda_i^0 = u_i(y_i)$$

The update law for λ becomes:

$$\lambda_i^{t+1} = \lambda_i^t - \beta_t \sum_{l \in N_i^-} \Big(\lambda_i^t - \lambda_l^t\Big) - \alpha_t \Big(x_i^t - y_i\Big)$$

$$x_i^{t+1} = C\Big(u_i^{-1}(\lambda_i^{t+1})\Big)$$

β_t is the "consensus" term, where the local value of λ is adjusted based on the values of the local nodes. $alpha_t$ is the "innocvation" term, that corrects according to the offset from the desired range.

The convergence criteria, in addition to previous assumptions, that:

$$\lim_{t\to\infty} \alpha_t = \lim_{t\to\infty} \beta_t = 0$$

$$\sum_{t=1}^{\infty} \alpha = \sum_{t=1}^{\infty} \beta = \infty$$

$$\lim_{t\to\infty} \frac{\beta_t}{\alpha_t} = \infty$$

This algorithm also claims convergence for cases of noisy data and communication losses. The requirement here is the the graph is connected *on average*. The noise for the estimate of p_i and x_i for a given time step is assumed to be zero-mead with a finite variance. Then the problem will converge given the requirement:

$$\sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

The Convergence rate for the main formula is not exponential with the given designs on the meta-parameters. However, the paper proposes that with a properly designed constant step size, the values will converge to a value withing the neighborhood of λ^* with at an exponential rate. Define the step sizes such that:

$$\beta = \kappa\gamma, \text{ and } \alpha = \kappa$$

$$\gamma > \gamma_\epsilon$$

$$\kappa > 0, \text{ and sufficiently small}$$

Here γ_ϵ is defined as some lower bound on allowable values of γ to allow the algorithm to converge within an error ϵ given by

$$\|\lambda^\epsilon - \lambda^*\| \leq \epsilon$$

The convergence rate is then bounded by:

$$\|\lambda^t - \lambda^\epsilon\| \leq c_1 r_{\gamma\,\kappa}^t \|\lambda^0 - \lambda^\epsilon\|$$

$$c_1 > 0\,, \text{ and } 0 < r_{\gamma\,\kappa} < 1$$

2.4 Algorithm 3

The algorithm in [3] approaches the solution in a manner similar to the Consensus + Innovation algorithm. To begin we will introduced a variant of the normalized adjacency matrix. Define a matrix A^+ and A^-such that:

$$a_{i,j}^+ = \begin{cases} \frac{1}{d_{n,i}^+}, & \text{if } (i,j) \in E_n. \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

$$a_{i,j}^- = \begin{cases} \frac{1}{d_{n,i}^-}, & \text{if } (i,j) \in E_n. \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

These are different than R as they are weighted instead by the degree of the node corresponding to each row rather than each column, and are row-stochastic. This matrix, rather than R or Q from before, is used because it has different desired properties for the collective estimation of the supply-demand mismatch, which will be addressed later. As before, this paper will discuss the case where each generator starts with an estimated of the virtual demand y_i using algorithm 0, as that is the implementation used in the next section.

However, it can also be extended to the case of computing over the whole graph, where non-generating nodes have a fixed value for x_i and some estimate of the virtual demand.

Initialize:

$$\lambda_i^0 = \text{ any feasible guess}$$

$$x_0 = C\Big(u_i^{-1}(\lambda_i^0)\Big)$$

$$z_i^0 = y_i^0 - x_i^0$$

The update is then:

$$\lambda_i^{t+1} = \sum_{j \in N_i^+} a_{ij}^+ \lambda_j^t + \epsilon_i z_i^t$$

$$x_{t+1} = C\Big(u_i^{-1}(\lambda_i^{t+1})\Big)$$

$$z_i^{t+1} = \sum_{j \in N_i^+} a_{ij}^- z_j^t - (x_i^{t+1} - x_i^t)$$

Which converges asymptotically for some sufficiently small $\epsilon > 0$. This algorithm relies on similar principles to [2], where there is iterative convergence based on metrics of local consensus of the λ terms. However, the structure is different where the structure of the graph where the terms are weighted by their relative degree, rather than a decreasing step size. the term z_i^t is structured in a way so that:

$$\sum_{i \in V_n} x_i^{t+1} + z_i^{t+1} = \sum_{i \in V_n} x_i^t + z_i^t$$

Plugging in the definitions of z_i^0, we can rewrite this to see:

$$\sum_{i \in V_n} z_i^t = P^* - \sum_{i \in V_n} x_i^t \quad \forall\, t \geq 0$$

Thus this equation can be thought of as a form of distributed proportional control, where ϵ is the control gain on the system.

3 Implementation

This section deals with implementing these algorithms on a model system. For this problem we will assume all generators have a convex cost function of the form:

$$f_i(x) = \frac{(x - \alpha_i)^2}{2\beta_i}$$

This is commonly done as it simplifies the form of the solution to the dual problem, as we can write:

$$u_i(x) = \frac{x - \alpha_i}{\beta_i}$$

$$u_i^{-1}(\lambda) = \lambda\beta_i + \alpha_i$$

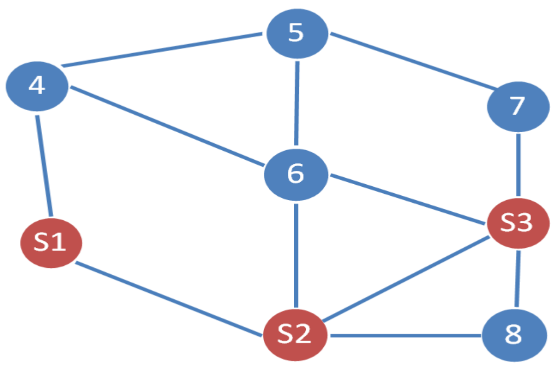

Figure 1: Topology of the Model Graph

To define the values in the table the associated graph is decomposed into two parts: $G_m \setminus G_n$ and G_n. G_n, the graph of all generation nodes is described below:

Index	α	β	$\underline{x}$	$\bar{x}$
1	-1	3	0	2.1
2	-1	2	0	1.0
3	-1	2	0	5.0

The graph of all non-generating nodes with demand is given by:

Index	p_i
4	1
5	1.4
6	1.1
7	.9
8	.2

These values were chosen for simplicity in seeing the validity of the solution. Generator 1 has a lower cost function than the other two, so it should have a higher set value. Generators 2 and 3 should have the same value in the unconstrained case, but generator two was adjusted so that this optimal value is higher than the upper bound, whereas generator 3 does not have this issue. The total demand for the grid is 4.1.

For these experiments, Algorithm 0 was first run in all 3 cases. This was done to ensure that this mapping remained valid, as it is a proposition only given in [5], but was extended into the cases of 2 and 3. For all cases, the convergence criteria for the steps in algorithm 0 where:

$$|k_i^{t+1} - k_i^t| \leq 10^{-6}$$

Number of steps for convergence of p^*, s^*, and y^* were 29, 27, and 19, respectively. All algorithms were run with a stopping criteria of:

$$|\sum x_i^t - P^*| < .0013$$

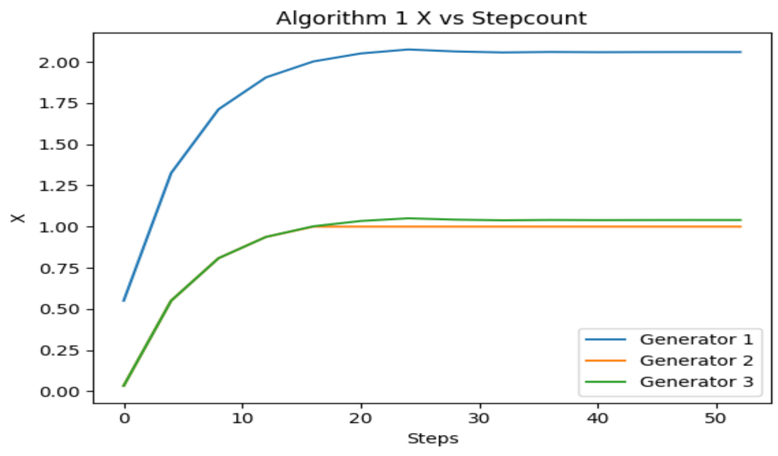

Figure 2: Progression of x_i^k over time. Convergence took 52 iterations over z_i^t total and 14 bisection steps. Final cost was 3.601

This was chosen at it corresponds to the convergence of [5] after 14 bisection steps.

This algorithm relies on 2 different levels of steps: the bisection step and the iterations on z within each bisection step. The values in figure 2 are scaled x values over the number of z iterations, although only 14 x values were calculated. The strength in this algorithm is in the ease of use - it requires no meta-parameters, is fully distributed, and allows for no sharing of private information. It also has a low-computational cost for graphs where most nodes are not generating power. However, it lacks the flexibility of the other algorithms in that is is unable to perform computations without using algorithm 0, which may be simpler in the event of different graph topology. Additionally, it makes the assumption that each node is aware of the overall diameter of the graph in order to determine when bisection occurs, which may not be a valid assumption in all cases.

Algorithm 2 proved to be unreliable. Although the paper claims that strong connectivity is sufficient for the algorithm to converge, the proper values for α_t and β_t didn't converge to the proper values after doing a line search within a limit of 10000000 iterations. However, a modified version of the graph where G_n is a complete graph (i.e. Node 1 and Node 3 are connected) was required to find a proper convergence. In this case, the parameters were defined as:

$$\alpha_t = \frac{.004}{t^{.01}}$$

$$\beta_t = \frac{.2}{t^{.0001}}$$

Convergence was found within 2063 iterations in this case.

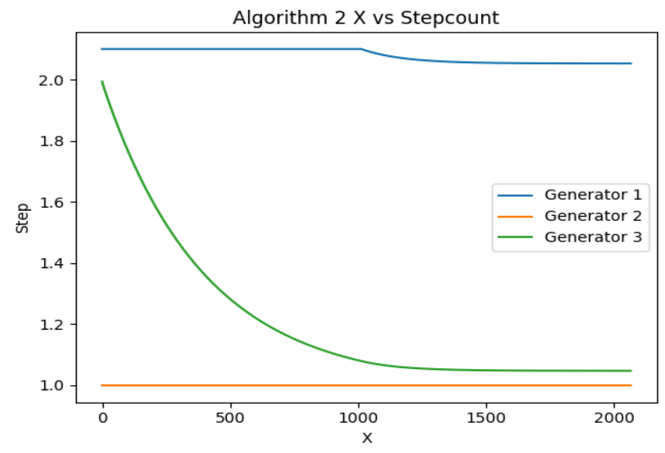

Figure 3: Progression of x_i^k over time for algorithm 2. Convergence took 2063 iterations total.

Convergence using the original strongly connected graph was difficult as it relied on tuning the parameters in the correct ratio so that $lambda_i$ converged to a single value, which is required for an optimal solution. Certain choice of constant step size where $\alpha > \beta$ allowed for the algorithm to converge to a steady state cost where each node had a different steady-state value of λ_i, at which supply met demand. For the values of: $\alpha = 1/1000$, $\beta = 1/18000$, the algorithm converged to $x_1 = 1.505$, $x_2 = 1$, and $x_3 = 1.505$ for nodes 1, 2, and 3, respectively. This solution corresponded to a local minimum cost for the case where λ didn't converge to a single value.

To attempt to find proper convergence, 1600 parameter combinations were tested. Each metaparameter was described in the form:

$$\alpha_t = \frac{a}{t^c}$$

$$\beta_t = \frac{b}{t^d}$$

Where the values of a, b, and c corresponded to 20 values spaced linearly between .001 and .1 space linearly. The 20 values of d were chosen between .0001 and .01, spaced linearly. These ranges were chosen based on manual pre-tuning of parameters around points where the algorithm solution tended to vary between instability, and sub-optimal convergence. No solutions were found.

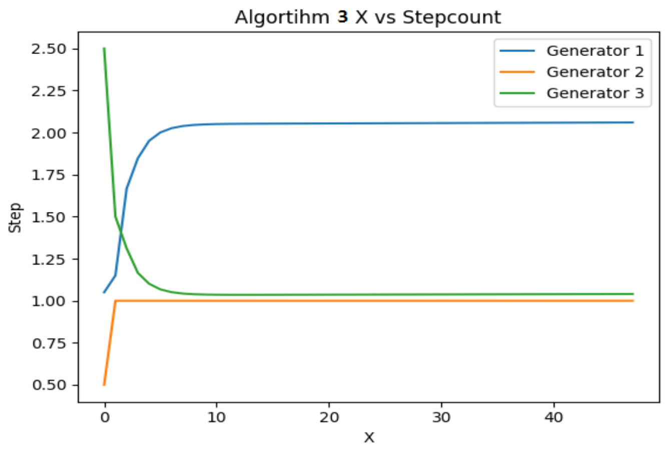

Figure 4: Progression of x_i^k over time for algorithm 3. Convergence took 48 iterations total.

The algorithm from [3] was run. ϵ was chosen as $\frac{1}{19500}$. Convergence for this case took 48 iterations, and converged to the same solution as in the bisection method.

Recall that this algorithm was run from the case where the virtual demand y was calculated from algorithm 0 first, and was run only over the 3 generator nodes. For comparison, the algorithm was also ran over the graph of all nodes, where the virtual demand at each node was set to be the initial demand. For this case, 22,144 iterations were required for convergence with the same ϵ.

It should be noted that while ϵ here was tuned to be a constant, there is no requirement on this. The trade off in the choice of ϵ for this case is similar to any normal gradient decent step size. A sufficiently small value of ϵ is needed for stability and convergence, whereas higher values tended to diverge. The choice of ϵ for this problem was based off of the highest single step size at which the solution converged. The design of using a single value vs unique values for each node thus also shows a trade off between programmer effort and optimal convergence criteria.

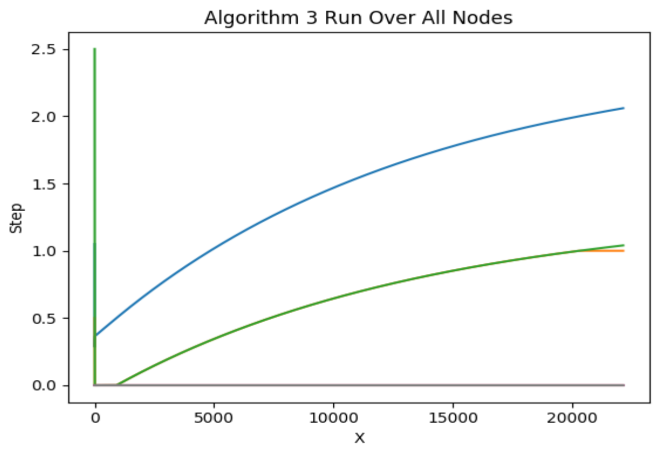

Figure 5: Progression of x_i^k over time for algorithm 3 for computing over the graph of all nodes. Convergence took 22,144 iterations total.

4 Conclusion

This paper introduces 3 algorithms from literature focusing on the distributed Economic Dispatch Problem. These algorithms were chosen for their similarity in approaches and requirements for convergence. Specifically, they all solve the solution using a primal-dual approach inspired by the analytic solution in the centralized case. Each node requires only information about it's own cost function and communication with nodes in it's local area. While a leader node is assumed for [2] and [3], these are shown to not be necessary by incorporating the consensus-based approach for finding an estimate of virtual demand from first running algorithm 0 over the net, which allows for reduced computational cost in the event that $|V_n| << |V_m|$.

Other algorithms exist for solving the economic dispatch problem. [4], an earlier text, introduces first order gradient descent and Lagrangian relaxation. Many algorithms rely on a centralized leader node, such as [6], or rely on nodes sharing their local cost functions. Other algorithms, such as [1], seek to solve a different problem, known as the welfare optimization, which considers the demand at each node as a different cost function to optimize over, rather than constraining the problem to require meeting a static demand.

Bibliography

[1] Q. Dong et al. "Distributed Demand and Response Algorithm for Optimizing Social-Welfare in Smart Grid". In: *2012 IEEE 26th International Parallel and Distributed Processing Symposium*. May 2012, pp. 1228–1239. DOI: `10.1109/IPDPS.2012.112`.

[2] S. Kar and G. Hug. "Distributed robust economic dispatch in power systems: A consensus x002B; innovations approach". In: *2012 IEEE Power and Energy Society General Meeting*. July 2012, pp. 1–8. DOI: `10.1109/PESGM.2012.6345156`.

[3] Sicong Tan, Shiping Yang, and Jian-Xin Xu. "Consensus based approach for economic dispatch problem in a smart grid". In: *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*. Nov. 2013, pp. 2011–2015. DOI: `10.1109/IECON.2013.6699440`.

[4] Allen J Wood and Bruce F Wollenberg. *Power generation, operation, and control*. John Wiley & Sons, 2012.

[5] H. Xing et al. "Distributed Bisection Method for Economic Power Dispatch in Smart Grid". In: *IEEE Transactions on Power Systems* 30.6 (Nov. 2015), pp. 3024–3035. ISSN: 0885-8950. DOI: `10.1109/TPWRS.2014.2376935`.

[6] Dayong Ye, Minjie Zhang, and Danny Sutanto. "Decentralised dispatch of distributed energy resources in smart grids via multi-agent coalition formation". In: *Journal of Parallel and Distributed Computing* 83 (2015), pp. 30–43. ISSN: 0743-7315. DOI: `https://doi.org/10.1016/j.jpdc.2015.04.004`. URL: `http://www.sciencedirect.com/science/article/pii/S0743731515000672`.