Learning Proximal Operators with Gaussian Processes

Truong X. Nghiem
School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Flagstaff, AZ, USA
truong.nghiem@nau.edu

Giorgos Stathopoulos and Colin N. Jones
Laboratoire d'Automatique
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
{georgios.stathopoulos, colin.jones}@epfl.ch

***Abstract*—Several distributed-optimization setups involve a group of agents coordinated by a central entity (coordinator), altogether operating in a collaborative framework. In such environments, it is often common that the agents solve proximal minimization problems that are hidden from the central coordinator. We develop a scheme for reducing communication between the agents and the coordinator based on learning the agents' proximal operators with Gaussian Processes. The scheme learns a Gaussian Process model of the proximal operator associated with each agent from historical data collected at past query points. These models enable probabilistic predictions of the solutions to the local proximal minimization problems. Based on the predictive variance returned by a model, representative of its prediction confidence, an adaptive mechanism allows the coordinator to decide whether to communicate with the associated agent. The accuracy of the Gaussian Process models results in significant communication reduction, as demonstrated in simulations of a distributed optimal power dispatch application.**

***Index Terms*—distributed optimization, communication, proximal operator, ADMM, Gaussian Process, machine learning**

I. Introduction

We consider setups that consist of a population of agents, typically (linear) dynamical systems that pursue local objectives and operate under constraints, and a coordinator. The coordinator pursues a system-wide objective in cooperation with the agents. In many such settings, the agents' data are supposed to remain private, either due to physical limitations or due to design choices. Consequently, such problems call for distributed solutions. Under some common structural assumptions, schemes like the Proximal Gradient Method (PGM) and the Alternating Direction Method of Multipliers (ADMM) are suitable choices for solving these problems (see [2] and the references therein).

In the majority of these schemes, the local subproblems that need to be solved are cast as *proximal minimization problems*, a term used to describe optimization problems that are regularized by the addition of a properly scaled quadratic term in their objectives [8]. In the course of the execution, the agents need to communicate the solutions to the proximal minimization subproblems to the coordinator, who will, in turn, manipulate the agents' objectives by broadcasting incentives that skew their local policies toward the global target. These schemes often require a large number of iterations and hence data exchanges between the agents and the coordinator. As a consequence, they often incur extensive communication, which might be highly undesirable in systems where communication is particularly expensive, in terms of *time* (e.g., delays and packet loss in weak networks), *energy* (e.g., battery- and solar-powered remote nodes in wireless sensor and actuator networks), or *cost* (e.g., performant and reliable networks require costly investment). Therefore, *it would be useful if the coordinator could learn to predict the proximal optimizers of its agents in order to skip, and hence reduce, some communication during the execution of these distributed optimization schemes*.

The idea of estimating the solution to a proximal minimization problem with the purpose of reducing communication overhead appeared in the authors' previous work [16], that was further developed in [15]. In the mentioned works, the authors estimate a convex set that contains the solution. Construction of the set is based on the theory of *the Moreau envelope function* (see, e.g., [10, Chapter 1.G]) and its important connections with the proximal operator. The structural properties of the Moreau envelope function allow the coordinator to cast the problem of estimating the proximal optimizer associated with an agent as the problem of estimating a gradient of the Moreau envelope function of that agent at a given point. Subsequently, the coordinator is able to bound the set of all possible optimizers to be within a convex set, explicitly described as the intersection of ellipsoids and iteratively refined every time that a communication round occurs.

In this work, we leverage the structural properties of the Moreau envelope function to locate the unknown optimizers by means of a different approach, using Gaussian Processes. This is made possible by the fact that the gradient estimation problem is a learning problem, which develops a model to predict the gradient of an unknown function from historical observations of the function value and its gradient. In other words, we learn the proximal operators associated with the agents using Gaussian Processes. Gaussian Processes are particularly chosen for their high flexibility, their tendency to work well with small data sets, and their ability to provide

This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 755445).

probabilistic confidence of their predictions. In our approach, Gaussian Process models are extended to incorporate derivative observations of the underlying Moreau envelope functions to enhance their accuracy and to predict the gradients at new inputs. We then devise a mechanism that, based on the predictive variance returned by the Gaussian Process model associated with an agent, decides whether the coordinator should communicate with that agent. Through simulations of a distributed optimal power dispatch application, our proposed method is shown to achieve a significant communication reduction of 62.2% from the exact ADMM method, and 34.2% from the method developed in [15]. However, convergence properties of the proposed method have not been formally established in this paper.

The paper begins with the problem definition and an overview of the structural approach proposed in [15], [16] in Section III. It then describes our approach for learning the proximal operators with Gaussian Processes and our communication decision mechanism in Sections IV and V, respectively. Simulation results are presented in Section VI. The paper concludes with some future directions in Section VII.

II. Related Work

In the context of communication avoidance in distributed optimization, the work [13] proposes a scheme for reducing communication rounds for a stochastic version of the iterative soft thresholding algorithm. In a similar but rather more generic manner, the authors in [11], [7] propose communication-efficient distributed-optimization frameworks for large-scale machine learning applications. Coming to learning optimization algorithms with better practical performance, the authors in [5] use neural networks in order to model algorithmic iterations of commonly used algorithms as policies. This allows them to tune (typically difficult-to-tune) parameters that are involved in the mentioned iterations via training and validation. The learned policies demonstrate competitive behavior when it comes to differentiable and unconstrained objectives, although in low dimensional settings. Our proposed method is fundamentally different since (i) it is applicable to general (usually constrained) proximal minimization problems, (ii) scales to relatively high-dimensional setups, and (iii) does not aim to achieve better algorithmic design, but rather to reduce the communication overhead, often at the expense of an increased number of iterations for convergence.

III. Estimating Proximal Operators in Distributed Convex Optimization

A. The problem

Consider an optimization problem involving N agents and a coordinator that takes the form

$$\text{minimize} \quad h(x) + \sum_{i=1}^{N} f_i(x_i) \ , \tag{1}$$

where $f_i : \mathbb{R}^{n_i} \mapsto \mathbb{R} \cup \{+\infty\},\ i = 1, \dots, N$ are arbitrary convex functions, private to the corresponding agents, and $h : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$, $n = \sum_{i=1}^{N} n_i$, is the convex objective of the coordinator.

In distributed optimization settings like the one above, the population of agents and the coordinator cooperate so as to solve problem (1). In order to achieve this, it is often the case that a *decomposition method* is applied, e.g., ADMM. Since certain agents' data is supposed to remain private in many such settings, each agent iteratively solves a proximal minimization problem of the form

$$\mathbf{prox}_{\gamma f_i}(z_i^k) = \underset{x_i \in \mathbb{R}^{n_i}}{\arg\min} \left\{ f_i(x_i) + \frac{1}{2\gamma} \|x_i - z_i^k\|^2 \right\} \ . \tag{2}$$

During the execution of the employed decomposition algorithm, a sequence of points $\{z_i^k\}_{k \in \mathbb{N}}$ at which the proximal minimization problem is solved is generated by the coordinator. These points correspond to the coordinator's intervention in order to steer the agents toward the minimizer of the (global) objective h. The response vector $\mathbf{prox}_{\gamma f_i}(z_i^k)$ is then transmitted from the agent to the coordinator. Such iterative schemes can be seen as *querying mechanisms*, where the coordinator broadcasts some value $\{z_i^k\}$ and the agent responds with another.

In setups where communication is expensive, the coordinator would like to avoid excessive querying of the agents, especially if the elicited information is not of importance. *The goal is to estimate, to the best possible accuracy, the optimizer of* (2) *for a given (arbitrary) sequence* $\{z_i^k\}_{k \in \mathbb{N}}$*, without having to solve the optimization, namely to 'guess' the solution of a (local) proximal minimization problem from the coordinator's perspective, i.e., without knowledge of the function* f_i*.*

B. A structural approach

Focusing on just one agent, and with a slight abuse of notation for the sake of generalizing, we drop the subscript i and the superscript k (we set $f = f_i$, $x = x_i$, $n = n_i$ and $z = z_i^k$ in comparison to (2)). The proximal minimization problem reads

$$\mathbf{prox}_{\gamma f}(z) = \underset{x \in \mathbb{R}^n}{\arg\min} \left\{ f(x) + \frac{1}{2\gamma} \|x - z\|^2 \right\} \ . \tag{3}$$

In the authors' previous work [15], we introduce some structure in (3) so as to estimate it. It turns out that the value function of (3), also known as the Moreau envelope of f and defined as

$$f^\gamma(z) = \min_{x \in \mathbb{R}^n} \left\{ f(x) + \frac{1}{2\gamma} \|x - z\|^2 \right\} \ , \tag{4}$$

is *convex and differentiable with Lipschitz continuous gradient with constant* $1/\gamma$ when f is a convex function. It is also known (see, e.g., [1, Proposition 5.1.7]) that *the unique solution to the proximal minimization* $x^\gamma(z) = \mathbf{prox}_{\gamma f}(z)$ *can be written as*

$$x^\gamma(z) = z - \gamma \nabla f^\gamma(z) \ , \tag{5}$$

i.e., it is the point at which the gradient iteration of f^γ*, evaluated at* z*, lands.*

It is evident from (5) that, given an arbitrary point z, the gradient $\nabla f^\gamma(z)$ is all that is needed in order to reconstruct the optimizer of (3). Therefore, the coordinator aims at constructing *a set of possible gradients at z and a way to evaluate the quality of the estimated gradient (how far away the estimate can be from the actual gradient) contained in the set*.

C. ADMM with estimated prox for distributed optimization

Back to solving (1), by introducing a copy $y = x$ and the associated dual variables λ, the corresponding ADMM iteration results in the updates

$$\begin{aligned} x^{k+1} &= \arg\min_{x\in\mathbb{R}^n} \left\{h(x) + (\rho/2)\|x - y^k + (1/\rho)\lambda^k\|^2\right\} \\ y_i^{k+1} &= \arg\min_{y_i\in\mathbb{R}^{n_i}} \left\{f_i(y_i) + (\rho/2)\|y_i - x_i^{k+1} - (1/\rho)\lambda_i^k\|^2\right\} \\ \lambda^{k+1} &= \lambda^k + \rho(x^{k+1} - y^{k+1}) \ , \end{aligned} \tag{6}$$

where $\rho > 0$ is the penalty parameter. The y-minimization step constitutes local proximal minimization problems of the form

$$\begin{aligned} y_i^{k+1} &= \mathbf{prox}_{\frac{1}{\rho}f_i}(\underbrace{x_i^{k+1} + (1/\rho)\lambda_i^k}_{v_i^k}) \\ &\overset{(5)}{=} v_i^k - \gamma\nabla f_i^\gamma(v_i^k), \quad \gamma = 1/\rho \ . \end{aligned} \tag{7}$$

The authors in [15] propose a method, henceforth called STEP (STructural Estimation of Proximal operator), to locate the unknown gradient $\nabla f_i^\gamma(v)$, for any point v (here $v = v_i^k$).

1) At algorithmic iteration k, the coordinator is given a number of query points $\{z_{i,j}\}_{j\in\mathcal{J}_i}$, namely a number of points at which $\mathbf{prox}_{\frac{1}{\rho}f_i}(z_{i,j})$ has been computed and transmitted to the coordinator by agent i. The set $\mathcal{J}_i$ contains the indices of queries generated up to the current algorithmic iteration, for every agent $i = 1, \dots, N$.
2) For each query point $z_{i,j}$ and any new point v the coordinator can explicitly form an ellipsoidal set $\mathcal{G}_i(v; z_{i,j})$, a set within which $\nabla f_i^\gamma(v)$ is located, given knowledge of $\nabla f_i^\gamma(z_{i,j})$.
3) By taking the intersection of the mentioned ellipsoidal sets, the coordinator keeps in its memory a set $\mathcal{G}_i(v) := \bigcap_{j\in\mathcal{J}_i}\mathcal{G}_i(v; z_{i,j})$ per agent. The set is updated whenever a communication round occurs, i.e., with every new query point, $\mathcal{G}_i(v)$ is augmented by one more (ellipsoidal) inequality.
4) At each iteration, the coordinator first makes a prediction regarding $\nabla f_i^\gamma(v)$. The prediction is based on looking in the pool of query points $\{z_{i,j}\}_{j\in\mathcal{J}_i}$ and finding the one, denoted by z_{i,j^*}, whose gradient is probably 'the closest' to the actual one, in some distance measure.
5) Finally, if $\nabla f_i^\gamma(z_{i,j^*}) \in \mathcal{G}_i(v)$, then $\nabla f_i^\gamma(v) = \nabla f_i^\gamma(z_{i,j^*})$, otherwise a projection onto $\mathcal{G}_i(v)$ is performed by means of solving the Quadratically Constrained Quadratic Program (QCQP)

$$\begin{aligned} \text{minimize} \quad & \|g_i - \nabla f_i^\gamma(z_{i,j^*})\|_2 \\ \text{subject to} \quad & g_i \in \mathcal{G}_i(v; z_{i,j}),\ j = 1, \dots, \mathcal{J}_i \ , \end{aligned} \tag{8}$$

with variable $g_i \in \mathbb{R}^{n_i}$.

In summary, the process consists of two main parts: steps 1-3, where a set containing the gradient is built by the coordinator, and steps 4-5, where a heuristic method is invoked to choose a point within the set.

Although the method exhibits good practical performance, there are two issues:

1) There is no clear guideline on how to choose a good approximation of the unknown gradient $\nabla f_i^\gamma(v_i^k)$. More importanly, although the exact gradients at the past query points are available, they are only used to bound the prediction of the gradient at the new query point.
2) The communication decision mechanism proposed in [15] is not based on any property of the proximal operator models nor of ADMM. In fact, this decision is algorithm-dependent (see [16] where another criterion for communication is designed for PGM).

We seek to address the above issues by: 1) developing proximal operator models that meaningfully use the past gradients in predicting the unknown gradient; and 2) devising a uniform communication test that exploits properties of these predictive models. In the rest of this paper, we present such an approach based on Gaussian Processes.

IV. Gaussian Processes for Estimating Proximal Operators

The previous section establishes a structural approach to estimate the proximal optimizer (3), more specifically the unknown gradient $\nabla f_i^\gamma(v_i^k)$, based on knowledge of the proximal optimizers at a finite set of past query points. This estimation problem can be formalized as follows.

Problem 1 (Gradient Estimation): For each agent i at algorithmic iteration k, given a set $\{(z_{i,j}, f_i^\gamma(z_{i,j}), \nabla f_i^\gamma(z_{i,j}))\}_{j\in\mathcal{J}_i}$ of query points and their corresponding Moreau envelope values and gradients, and a new query point v_i^k, predict the unknown gradient $\nabla f_i^\gamma(v_i^k)$ such that it lies inside $\mathcal{G}_i(v_i^k)$ defined above.

The gradient estimation problem is a machine learning problem, which aims to learn the gradient ∇f_i^γ from the observation data at the query points. In this paper, we propose to use *Gaussian Processes* (GP) for this learning problem. Readers who are unfamiliar with GPs are referred to the Appendix for a brief tutorial on GPs, and to [9] for a deep treatment of GPs for machine learning. GPs are chosen for several of their advantages (cf. the Appendix):

- GPs work well with small data sets: the number of query points for each agent is small, typically in the tens, hence the observation data set is also small. Some machine learning methods, for example neural networks, require a large data set to achieve good accuracy. GPs are known to work well with small data sets [4].
- GPs are highly flexible: because the structure of f_i^γ is unknown, coupled with the small size of the observation data, it is desirable to use a flexible machine learning method that does not impose a fixed input-output structure on the learned function. GPs satisfy

this requirement as they use covariance functions to specify a structural relationship between the function inputs instead. In addition, the Lipschitz continuity of the Moreau envelope implies that its value at an input is influenced by observation data at nearby points more than at points further away. This property aligns well with GPs and how the covariance function works.

- GP inference provides a variance: this estimate of the predictive confidence indicates how accurate a prediction is, and is particularly useful for deciding whether to communicate or not. Section V will discuss how the variance can be used as a communication criterion.

A straightforward approach to tackle the gradient estimation problem is to learn a multi-output GP of the function $\nabla f_i^\gamma : \mathbb{R}^{n_i} \mapsto \mathbb{R}^{n_i}$ from z_i directly to the gradient $\nabla f_i^\gamma(z_i)$. However, this approach fails to exploit the important fact that the outputs constitute the gradient of the Moreau envelope function, whose value is known, therefore losing valuable information to enhance its accuracy. Rather, *we propose to learn the Moreau envelope function $f_i^\gamma : \mathbb{R}^{n_i} \mapsto \mathbb{R}$ and incorporate the gradient into the GP model by taking advantage of a special property of GPs*.

A. Gaussian Processes with Derivative Observations

Because differentiation is a linear operation, the derivative of a GP is also a GP [12]. This special property of GPs enables using derivative observations of the underlying function in learning as well as predicting its derivatives at a new input. Consider an agent i and the GP model of its associated Moreau envelope f_i^γ. Similar to Section III-B, we will drop the subscript i, e.g., f^γ and z_j in place of f_i^γ and $z_{i,j}$. We will also use some notations related to GPs, introduced in the Appendix.

Let us denote $f_j^\gamma = f^\gamma(z_j)$ and $\nabla f_j^\gamma = \nabla f^\gamma(z_j)$ for query points z_j, $j \in \mathcal{J}$. Suppose that noise-free derivative observations $\nabla f_j^\gamma = \left[\frac{\partial f_j^\gamma}{\partial z_j^{(d)}}\right]_{d=1,\dots,n}$ are available for each z_j, where $z_j^{(d)}$ is the d-th element of z_j. They can be included in the augmented observation vector $\hat{y}_j^T = \left[f_j^\gamma, \left(\nabla f_j^\gamma\right)^T\right]$. Following [12], the covariance matrix is correspondingly extended, for any pair of query points $j, l \in \mathcal{J}$, with the covariances between the observations and the partial derivatives,

$$\operatorname{Cov}\left[\frac{\partial f_j^\gamma}{\partial z_j^{(d_j)}}, f_l^\gamma\right] = \frac{\partial}{\partial z_j^{(d_j)}} \operatorname{Cov}\left[f_j^\gamma, f_l^\gamma\right] = \frac{\partial}{\partial z_j^{(d_j)}} k\left(z_j, z_l\right),$$

and the covariances between the partial derivatives themselves,

$$\begin{aligned}\operatorname{Cov}\left[\frac{\partial f_j^\gamma}{\partial z_j^{(d_j)}}, \frac{\partial f_l^\gamma}{\partial z_l^{(d_l)}}\right] &= \frac{\partial^2}{\partial z_j^{(d_j)} \partial z_l^{(d_l)}} \operatorname{Cov}\left[f_j^\gamma, f_l^\gamma\right] \\ &= \frac{\partial^2}{\partial z_j^{(d_j)} \partial z_l^{(d_l)}} k\left(z_j, z_l\right).\end{aligned}$$

Here, $1 \le d_j, d_l \le n$ and $k(\cdot,\cdot)$ is the GP covariance function (cf. the Appendix). Most commonly used covariance functions, such as the Squared Exponential covariance function, are differentiable, so the above covariances are well-defined. We can now form the joint probability of the vectors $\hat{y}_j$, which are used for both learning the model, e.g., optimizing the hyperparameters, and predicting the gradient $\nabla f^\gamma(v^k)$ in addition to the output $f^\gamma(v^k)$ at a new query point v^k.

B. Learning Proximal Operators with Gaussian Processes

Back to the Gradient Estimation problem, we will utilize the full knowledge of the query points to learn a GP model that can predict the gradient $\nabla f_i^\gamma(v_i^k)$ at a new point v_i^k. Specifically, for each agent i, the coordinator will maintain a GP model, called a *proxGP*, which is trained on the output data $\{f_i^\gamma(z_{i,j})\}_{j\in\mathcal{J}_i}$ with derivative observations $\{\nabla f_i^\gamma(z_{i,j})\}_{j\in\mathcal{J}_i}$ corresponding to regression vectors $\{z_{i,j}\}_{j\in\mathcal{J}_i}$. In this paper, we choose for the proxGP the zero mean function, $\mu(x) = 0$, and the widely-used smooth Squared Exponential covariance function

$$k(x, x') \coloneqq \sigma^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{n} \frac{(x_d - x'_d)^2}{\lambda_d^2}\right),$$

where $\theta = [\sigma, \lambda_1, \dots, \lambda_n]^T$ are the hyperparameters.

The proxGP is used to predict $\nabla f_i^\gamma(v_i^k)$, represented by its mean $\mathbb{E}\left[\nabla f_i^\gamma(v_i^k)\right]$ and covariance matrix $\operatorname{Cov}\left[\nabla f_i^\gamma(v_i^k)\right]$. Although the full covariance matrix can be computed, we are only interested in the individual variance of each dimension, namely the diagonal elements of the covariance matrix: $\operatorname{Var}\left[\nabla f_i^\gamma(v_i^k)\right] = \operatorname{diag}\left(\operatorname{Cov}\left[\nabla f_i^\gamma(v_i^k)\right]\right)$. Because the proxGP model does not take into account the constraint set $\mathcal{G}_i(v_i^k)$, its predictive mean might fall outside the constraint and is invalid. In such cases, we follow the suggestion in [15] to project it onto $\mathcal{G}_i(v_i^k)$ by solving the QCQP (8).

Whenever the coordinator communicates with agent i to obtain the exact proximal optimizer at a new query point (see Section V for when such communication occurs), the proxGP model is updated, i.e., its hyperparameters are re-optimized, with the new data. This allows the proxGP model to refine its hyperparameters and improve its prediction accuracy.

V. Communication

In the previous section, we discuss learning the proximal operators of agents with GPs, which are used for solving the distributed optimization problem (1) with ADMM as presented in Section III-C. *A key challenge, that remains unsatisfactorily addressed in [15], is a certification test based on which the coordinator decides to communicate with an agent or not.* This section will propose a certification test that exploits the predictive variance returned by a proxGP model.

The variance $\operatorname{Var}\left[\nabla f_i^\gamma(v_i^k)\right]$ returned by the proxGP model for agent i is a measure of the uncertainty, or confidence, of the gradient prediction. The smaller the variance is, the more confident the model is in its prediction accuracy. Therefore, it is justifiable to use the variance to decide whether communication with an agent is warranted. We propose the following heuristic criterion: if $\max\left(\operatorname{Var}\left[\nabla f_i^\gamma(v_i^k)\right]\right) > \left(\epsilon_i^k\right)^2$ then communication is needed, otherwise it is not. In other words,

if the largest standard variance exceeds the varying threshold $\epsilon_i^k \geq 0$ then the coordinator will communicate with agent i.

An important design decision in the test is how to adapt the threshold ϵ_i^k. Empirically, we find that a constant threshold does not work well because, most likely, communication will either happen at every iteration (degeneration to exact ADMM) or rarely happen, causing ADMM to never converge or converge too slowly. Intuitively, the threshold should decrease gradually to account for the increased accuracy needed for the solution to converge. We find that a simple decaying threshold $\epsilon_i^k = \epsilon_i^0 r_i^k$ with a decay rate $0 < r_i < 1$ works well in low-dimensional cases, but often results in excessive communication in other cases.

Instead, we devise an adaptive mechanism, inspired by the communication test proposed in [15]. Let $s^k = \|x^k - y^k\|$ be the ADMM residual with respect to the consensus condition. Although monotonic decrease of the residual is neither a necessary nor a sufficient condition for the convergence of ADMM, it can be used to assess the *quality of the estimated proximal minimizer*. Let $\mathcal{C}^k$ be the set of agents with which the coordinator has communicated during algorithmic iteration k. After the iteration, the coordinator updates the thresholds ϵ_i of the agents following these rules:

- If $s^k < s^{k+1}$ (estimation quality is unsatisfactory), the thresholds of agents who did not communicate ($i \notin \mathcal{C}^k$) will be tightened by r_i, the other thresholds do not change;
- If $s^k \geq s^{k+1}$ (estimation quality is satisfactory), the thresholds of agents who did communicate ($i \in \mathcal{C}^k$) will be loosened by a fixed rate $\tau_i > 1$ (the rational is that the accuracy of the proxGP models of these agents is probably good enough but their thresholds are low), the other thresholds do not change.

Putting everything together, the reduced-communication distributed optimization algorithm based on GPs, called *STEP-GP*, is described in Algorithm 1. All steps are performed by the coordinator, except those with the comment "Agent i" are performed by the corresponding agent.

Algorithm 1 STEP-GP: Distributed Optimization with Estimated Proximal Operator based on Gaussian Processes

Require: $x_i^0 \in \mathbb{R}^{n_i}$, $\epsilon_i^0 > 0$, $r_i \in (0,1)$, $\tau_i > 1$, $\mathcal{G}_i = \emptyset$, $i = 1,\ldots,N$, $\rho > 0$, $k_{\text{stop}} > 0$

for $k = 0, 1, \ldots, k_{\text{stop}}$ **do**
  $\mathcal{C}^k \leftarrow \emptyset$
  Compute x^{k+1} from (6)
  $v^k \leftarrow x^{k+1} + (1/\rho)\lambda^k$
  for each agent i **do**
    Predict $g_i^k \leftarrow \mathbb{E}\left[\nabla f_i^\gamma(v_i^k)\right]$
    Solve (8) to project g_i^k if $g_i^k \notin \mathcal{G}_i(v_i^k)$
    if $\max\left(\text{Var}\left[\nabla f_i^\gamma(v_i^k)\right]\right) > \left(\epsilon_i^k\right)^2$ **then**
      $\mathcal{C}^k \leftarrow \mathcal{C}^k \cup \{i\}$
      Send v_i^k to Agent i
      Compute y_i^{k+1} from (6) ▷ Agent i
      Send $\left(y_i^{k+1}, f_i^\gamma(v_i^k)\right)$ to Coordinator ▷ Agent i
      Update data set of query points of Agent i
      Retrain proxGP of Agent i with new data
    else
      $y_i^{k+1} \leftarrow v_i^k - (1/\rho)g_i^k$
    end if
    Compute λ_i^{k+1} from (6)
  end for
  $s^{k+1} \leftarrow \|x^{k+1} - y^{k+1}\|$
  if $s^k < s^{k+1}$ **then**
    $\epsilon_i^{k+1} \leftarrow r_i \epsilon_i^k$ for all $i \notin \mathcal{C}^k$
  else
    $\epsilon_i^{k+1} \leftarrow \tau_i \epsilon_i^k$ for all $i \in \mathcal{C}^k$
  end if
end for

VI. Application: Optimal Exchange

We consider the exact same application used in [15] for comparison between our proposed GP-based approach and the approach presented therein. We first summarize the application problem and data, followed by a description of the implementation of our approach, simulation results, and a short discussion of the results.

We are interested in cooperatively tracking a reference power profile with a population of controllable buildings (CB). Our aim is to render a distribution feeder dispatchable, namely to use the CBs in order to achieve virtually perfect dispatchability of a set of devices consisting of uncontrollable loads and distributed generation. The idea comprises two stages, namely a day-ahead prediction of a dispatch plan and real-time operation, broken further into a high-level and a faster low-level controller, both model-based [14]. Our interest is to solve the high-level Model Predictive Control (MPC) problem that takes the form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(p_i) \\ \text{subject to} \quad & \sum_{t=1}^{T}\sum_{i=1}^{N}(p_i(t) - \hat{p}_i(t)) = r(t) \ . \end{aligned} \tag{9}$$

The variable $p_i(t)$ refers to the total amount (electrical equivalent) of the thermal consumption of the i^{th} controllable building at time instant t, where $i = 1,\ldots,N$, and $p_i = (p_i(0),\ldots,p_i(T-1)) \in \mathbb{R}^T$, $p = (p_1,\ldots,p_N) \in \mathbb{R}^{NT}$. The reference power profile is denoted by $r(t)$, while time spans from $t = 0,\ldots,T-1$, i.e., we have a T-timesteps ahead prediction of the power profile. The buildings can participate in the ancillary service market by increasing or decreasing their consumption with respect to some baseline power profile $\hat{p}_i$. Finally, the equality constraint enforces that the total power contribution equates the reference power profile.

The setup comprises small and medium office buildings, generated by the OpenBuild software [3]. The individual components $f_i(p_i) : \mathbb{R}^T \mapsto \mathbb{R} \cup \{+\infty\}$, $i = 1,\ldots,N$ are convex functions and correspond to local performance

criteria constrained inside convex polytopes. These polytopes ensure that *the power trajectories* $p_i,\ i = 1, \ldots, N$ *are feasible, i.e., they satisfy the buidlings' dynamics and will not violate any comfort constraint associated to the buildings' zonal temperatures.* More details about the exact form of these functions are given in [15].

As a final step, we perform the simple reformulation suggested in [2, Section 7.3.2], and the dispatchability problem (9) reduces to the *optimal exchange* problem, the ADMM iterations of which are given below:

$$\begin{aligned} p_i^{k+1} &= \underset{p_i \in \mathbb{R}^T}{\arg\min} \left\{ f_i(p_i) - \langle p_i, \lambda^k \rangle + \tfrac{\rho}{2} \|p_i - (p_i^k - \bar{p}_i^k)\|_2^2 \right\}, \\ & \quad i = 1, \ldots, N \\ \lambda^{k+1} &= \lambda^k + \rho(\bar{p}^{k+1} - r/N) \ , \end{aligned} \tag{10}$$

where $\lambda \in \mathbb{R}^{NT}$ are dual variables associated with the equality constraint in (9), while $\bar{p} = \frac{1}{N} \sum_{i=1}^N p_i$. We can now solve (9) by iterating the scheme (10).

A. Simulation Implementation

As in [15], the goal is to assign one-hour reference tracking to a mix of $N = 50$ buildings, 28 small and 22 medium-sized. The horizon length for the MPC problem is set to one day, i.e., $T = 24$.

Problem (9) is solved with three different methods:

1) *Exact*: this method uses ADMM with exact proximal operator computation in (6), which simplifies to (10) with $\rho = 20$;
2) *STEP*: the method proposed in [15] and detailed in Section III;
3) *STEP-GP*: the GP-based method proposed in this paper. For the communication test parameters (cf. Section V), we choose the initial threshold $\epsilon_i^0 = 0.012$ and the rates $r_i = 0.98$, $\tau_i = 1.05$ for all i.

Our focus in comparing these methods is the total number of communications between the coordinator and the agents, which indicates the ability of the latter two methods in reducing communication. To eliminate the difference in termination between them, the termination condition of each method is selected to be the distance between the iterate and the optimizer, namely $\|p^k - p^*\|/\|p^*\| \leq 10^{-3}$.

The simulation is implemented in MATLAB. The proximal minimization problems are modeled using the YALMIP toolbox [6] and solved with the Gurobi solver. For GP training and inference, we use the GPstuff toolbox [17]. GPstuff supports derivative observations in both hyperparameter optimization and inference, allowing us to query the mean and variance of any first derivative of the regression function, or the mean and full covariance matrix of the gradient, at a new input. Inspired by the heuristic technique used in [15] to reduce the computation time of each agent, we limit the number of quadratic constraints to the $M = 5$ most recent ones in solving the projection given by the QCQP (8). It was shown in [15] that this heuristic could save considerable computation time with little impact on the performance.

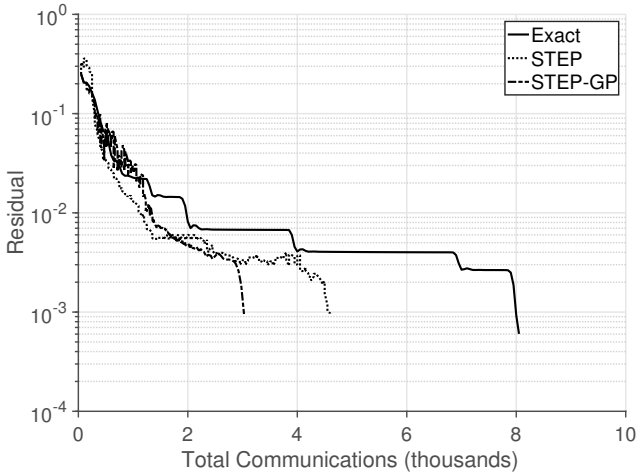

Fig. 1. Performance in terms of communication savings.

B. Simulation Results

a) Communication reduction: Fig. 1 depicts the number of total communications among all agents for the residual being computed as $\|p^k - p^*\|/\|p^*\|$. In comparison to the *Exact* ADMM implementation, the *STEP* method in [15] achieves approximately 42.5% reduction in communication, while the *STEP-GP* method proposed in this paper is able to reduce the communication even further, by approximately 62.2%. Compared to the *STEP* method, the *STEP-GP* method requires 34.2% fewer total communications of all agents, from $4,600$ down to $3,026$. Although not depicted here, the numbers of iterations required by both approximate methods increase from the baseline *Exact* method. However, interestingly, the *STEP-GP* method needs fewer iterations than the *STEP* method, 219 to 249, to achieve the same residual.

While in the *Exact* and *STEP* methods, all agents either communicate or do not communicate in an iteration (i.e., all-or-none), agents in our *STEP-GP* method decide their communication need individually, based on the prediction accuracy of each agent's proxGP model. Fig. 2 illustrates the communication need, represented by the accumulated number of communication rounds, of a typical small building agent and a typical medium-sized building agent. Observe that the small agent needs less communication than does the medium-sized agent.

b) Prediction accuracy: Fig. 3 depicts the prediction error of the proxGP model for a small building agent in non-communicating iterations (i.e., iterations in which the agent and the coordinator did not communicate). The error is defined as the ℓ_2-norm of the difference between the predicted gradient and the actual gradient $\nabla f_i^\gamma(v_i^k)$ as was computed by the proximal operator (see (7)). Observe that the error tends to decrease exponentially. It peaks at 0.078 and reaches 6.37×10^{-4} at the end. We do not have the prediction accuracy results of the *STEP* method.

c) Computation effort: Because the *STEP* and *STEP-GP* methods involve different computation steps in each iteration

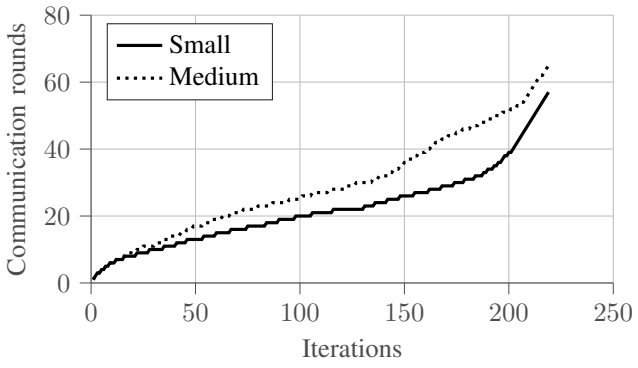

Fig. 2. Communication need of each building type.

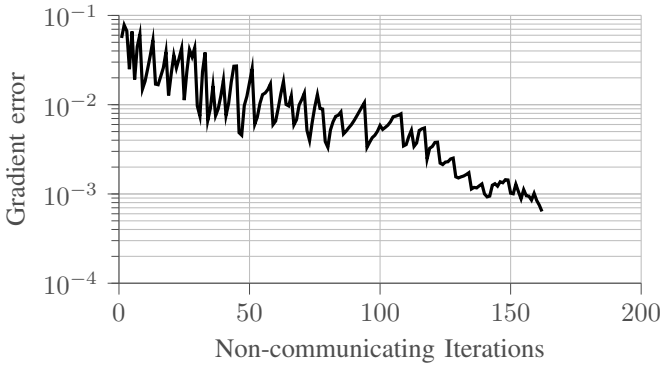

Fig. 3. Gradient prediction error of the GP model for a small building agent in non-communicating iterations.

for each agent, it is uninformative to compare their total computation time or average computation time per iteration. Instead, we will break down and compare the computation time for each major step in each iteration. In both methods, each iteration for each agent includes a gradient prediction step, where the gradient $\nabla f_i^\gamma(v_i^k)$ is predicted, followed by an optional projection step in case the predicted gradient falls outside the set $\mathcal{G}_i(v_i^k)$. In the *STEP-GP* method, after each communication between the coordinator and an agent, the proxGP model associated with the agent must be updated, i.e., its hyperparameters are re-optimized.

Table I summarizes the average computation time for these major steps in each iteration of the *STEP* and *STEP-GP* methods, and is explained below.

1) The gradient prediction step always happens in each iteration and for each agent. The *STEP* method uses a past gradient that is closest to the constraint set $\mathcal{G}_i(v_i^k)$; this operation is simple and takes negligible time. In the *STEP-GP* method, the proxGP model of an agent takes an average of 0.13 seconds to predict the gradient.
2) The projection step takes an average of 0.58 seconds for both methods. However, this step only occurs if the predicted gradient in the previous step falls outside $\mathcal{G}_i(v_i^k)$. The average chance that this step occurs in each iteration for each agent is 44.3% for the *STEP* method and slightly lower at 43.8% for the *STEP-GP* method.
3) The last step updates the prediction model for an agent after a communication round between it and the coordinator. The *STEP* method takes a negligible amount of time to simply record the data of the new query point. The *STEP-GP* method needs to re-optimize the hyperparameters of the associated proxGP model given the new data, which takes an average of 0.62 seconds. Note that this step does not always take place in each iteration but only after a communication round. The less communication is required, which is the goal of our proposed method, the less time is spent on this step.

TABLE I
AVERAGE TIME OF MAJOR STEPS IN EACH ITERATION.

Computation step	**STEP** method	**STEP-GP** method
1. Gradient prediction	negligible	0.13s
2. Projection	0.58s	0.58s
	Probability: 0.4429	Probability: 0.4376
3. Update after comm.	negligible	0.62s

It is worth pointing out that GPstuff, the GP toolbox we use, focuses on ease of use and features rather than computational efficiency, hence there is room for improvement.

Overall, the *STEP-GP* method is more complex than the *STEP* method, and therefore tends to take more computation time on average. However, it achieves more significant reduction in communication. In applications where communication reduction is critical while computation is affordable, for example where agents communicate through a slow and failure-prone cellular network, the advantage of the *STEP-GP* method is obvious.

VII. CONCLUSION

Distributed optimization algorithms in multi-agent setups often suffer from significant inherent communication overhead by requiring large numbers of data exchanges between the agents and the central coordinator. Built upon a previous framework for communication reduction in distributed optimization, we propose an approach in which the coordinator gradually learns the unknown proximal operators of the agents with Gaussian Process models and triggers communication with the agents based on the accuracy of their predictions. We also devise an adaptive communication test that exploits the predictive variance returned from these models. Simulations of an optimal power dispatch application show that our method achieves significant communication reduction in comparison to the exact ADMM algorithm and the previous framework. Our method will have a clear advantage in distributed optimization applications where communication reduction is critical to the performance of the systems.

The results in this work are based on empirical simulations and are not yet backed by theoretical results. A future goal is to refine the algorithm to have provable convergence property. We will also investigate ways to improve the accuracy of

the machine learning model by incorporating the convexity constraint of the Moreau envelope and the feasible gradient set constraint into the model. We plan to generalize the method to decentralized settings, for example in wireless sensor and actuator networks where only one-hop communication is allowed, which will broaden the applicability and practicality of our approach. On a different note, the proposed scheme can be used to construct a model for the decision-making process of an agent without the latter's consent. One could, therefore, imagine developing the method for preference elicitation in the context of mechanism design.

REFERENCES

[1] Dimitri P Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 2011.

[3] T. T. Gorecki, F. A. Qureshi, and C. N. Jones. Openbuild : An integrated simulation environment for building control. *2015 IEEE Conference on Control Applications (CCA)*, pages 1522–1527, 2015.

[4] Juš Kocijan. *Modelling and control of dynamic systems using Gaussian process models*. Springer, 2016.

[5] K. Li and J. Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

[6] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proc. of the CACSD Conference*, Taipei, Taiwan, 2004.

[7] Chenxin Ma, Jakub Konecný, Martin Jaggi, Virginia Smith, Michael I. Jordan, Peter Richtárik, and Martin Takác. Distributed optimization with arbitrary local solvers. *Optimization Methods Software*, 32(4):813–848, July 2017.

[8] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[9] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[10] R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational Analysis*. Springer Verlag, Heidelberg, Berlin, New York, 1998.

[11] Virginia Smith, Simone Forte, Chenxin Ma, Martin Takác, Michael I. Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *arXiv preprint arXiv:1611.02189*, 2016.

[12] Ercan Solak, Roderick Murray-Smith, William E Leithead, Douglas J Leith, and Carl E Rasmussen. Derivative observations in gaussian process models of dynamic systems. In *Advances in neural information processing systems*, pages 1057–1064, 2003.

[13] S. Soori, A. Devarakonda, J. Demmel, M. Gurbuzbalaban, and M.M. Dehnavi. Avoiding Communication in Proximal Methods for Convex Optimization Problems. *arXiv preprint arXiv:1710.08883*, 2017.

[14] F. Sossan, E. Namor, R. Cherkaoui, and M. Paolone. Achieving the Dispatchability of Distribution Feeders Through Prosumers Data Driven Forecasting and Model Predictive Control of Electrochemical Storage. *IEEE Transactions on Sustainable Energy*, 7(4):1762–1777, 2016.

[15] G. Stathopoulos and C. N. Jones. Communication reduction in distributed optimization via estimation of the proximal operator. *arXiv preprint arXiv:1803.07143*, 2018.

[16] G. Stathopoulos and C. N. Jones. A coordinator-driven communication reduction scheme for distributed optimization using the projected gradient method. In *Proceedings of the 17th IEEE European Control Conference, ECC 2018, Limassol, Cyprus*, 2018.

[17] Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen, Pasi Jylänki, Ville Tolvanen, and Aki Vehtari. GPstuff: Bayesian modeling with gaussian processes. *Journal of Machine Learning Research*, 14:1175–1179, 2013.

APPENDIX
A BRIEF TUTORIAL ON GAUSSIAN PROCESSES

This section is a brief introduction to Gaussian Processes (GP). For an in-depth treatment of GP, interested readers are referred to the excellent book [9] by Rasmussen and Williams.

Consider noisy observations y of an underlying function $f : \mathbb{R}^n \mapsto \mathbb{R}$ through a Gaussian noise model: $y = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$ and $x \in \mathbb{R}^n$. A GP of y is essentially a probability distribution on the observations of all possible realizations of f. By conditioning the GP on a finite set of observation data of f, one can derive the posterior distribution, which allows probabilistic inference at a new input. The formal definition of a GP can be found in [9] and is reproduced below.

Definition 1: A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

A GP of y is fully specified by its mean function $\mu(x)$ and covariance function $k(x, x')$,

$$\mu(x;\theta) = \mathbb{E}[f(x)]$$
$$k(x,x';\theta) = \mathbb{E}[(f(x)-\mu(x))(f(x')-\mu(x'))] + \sigma_n^2\delta(x,x')$$

where $\delta(x, x')$ is the Kronecker delta function. The mean and covariance functions, hence the GP, are parameterized by the *hyperparameter* vector θ, denoted by $y \sim \mathcal{GP}(\mu, k; \theta)$. There exists a wide range of covariance functions and combinations to choose from [9].

Given the regression vectors $X = [x_1, \ldots, x_N]^T$ and the corresponding observed outputs $Y = [y_1, \ldots, y_N]^T$, we define training data by $\mathcal{D} = (X, Y)$. The distribution of the output $y_\star$ corresponding to a new input vector $x_\star$ is a Gaussian distribution $\mathcal{N}\left(\bar{y}_\star, \sigma_\star^2\right)$, with mean and variance given by

$$\bar{y}_\star = \mu(x_\star) + K_\star K^{-1}(Y - \mu(X))$$
$$\sigma_\star^2 = K_{\star\star} - K_\star K^{-1} K_\star^T,$$

where $K_\star = [k(x_\star, x_1), \ldots, k(x_\star, x_N)]$, $K_{\star\star} = k(x_\star, x_\star)$, and K is the covariance matrix with elements $K_{ij} = k(x_i, x_j)$.

In theory, hyperparameters θ are random variables, which define the GP distribution on the function and whose posterior distributions are obtained by conditioning them on $\mathcal{D}$ using the Bayes' theorem. In practice, θ are often obtained by maximizing the likelihood: $\arg\max_\theta \Pr(Y|X, \theta)$.

The covariance function $k(x, x')$ indicates how correlated the outputs are at x and x', with the intuition that the output at an input is influenced more by the outputs of nearby inputs in the training data $\mathcal{D}$. In other words, a GP model specifies the structure of the covariance matrix of, or the relationship between, the input variables rather than a fixed structural input–output relationship. It is therefore *highly flexible* and *can capture complex behavior with fewer parameters*. A GP also provides an *estimate of uncertainty or confidence* in the predictions through the predictive variance. While the predictive mean is often used as the best guess of the output, the variance can be used in a meaningful way, as we show in this paper. In addition, GPs generally *work well with small data sets*, which is useful for any learning application where data are not easily obtained. Finally, GPs allow including prior knowledge of the underlying function by defining priors on the hyperparameters or constructing a particular structure of the covariance function. This feature enables *incorporating domain knowledge* into the GP model to improve its accuracy.