Interior Point Decomposition for Multi-Agent Optimization

Altuğ Bitlislioğlu * Ivan Pejcic + Colin Jones *

* Laboratoire d’Automatique, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
(e-mail: altug.bitlislioglu - ivan.pejcic - colin.jones @ epfl.ch).

Abstract: In this paper we present the application of the interior-point decomposition (IPD) method, which was originally formulated for stochastic programming, to optimization problems involving multiple agents that are coupled through constraints and objectives. IPD eliminates the need to communicate local constraints and cost functions for all variables that relate to internal dynamics and objectives of the agents. Instead, by using embedded barrier functions, the problem is solved in the space of coupling variables, which are in general much lower in dimension compared to internal variables of individual agents. Therefore, IPD contributes to both problem size reduction as well as data hiding. The method is a distributed version of the primal barrier method, with locally and globally feasible iterations and faster convergence compared to first-order distributed optimization methods. Hence, IPD is suitable for early termination in time-critical applications. We illustrate these attractive properties of the IPD method with a distributed Model Predictive Control (MPC) application in the context of smart-grids, where a collection of commercial buildings provide voltage support to a distribution grid operator.

Keywords: Distributed control, Predictive control, Multi-agent, Optimal control, Distributed optimization, Decomposition, Smart power applications, Demand response.

1. INTRODUCTION

Multi-agent optimization is receiving increased attention from the academy and industry with the rising prospect of smart grid applications. ‘Smart’ operation of the future’s electricity grid will involve coordination of various large and small players such as generators, consumers and grid operators. Efficient coordination of these complex agents with various dynamics, constraints and interests is an inherently difficult task and introduces many challenges. For example in a distribution grid, one can consider a demand response scheme where buildings, factories, and local generation facilities such as solar panels and generators are coordinated by a central agent, that is responsible for optimizing the aggregate power consumption/generation output and security of the grid. In this setting, it is desirable for the central agent to establish a consistent interface with local agents and consider only variables which affect the grid and aggregate objectives directly, whereas for commercial and residential agents, it is important to protect private data, such as internal states, constraints and objectives. Another requirement arises when optimization of the aggregate response is carried out in real-time. Due to time constraints for taking a decision, an algorithm should either terminate in a predetermined final time, or be able to terminate prematurely while having a feasible sub-optimal decision at hand.

In this paper, we consider the requirements of local data hiding, and anytime termination of optimization algorithms in time critical applications. We show that the interior-point decomposition (IPD) method (Zhao, 2001; Mehrrotra and Özçem, 2009b) satisfies both of these requirements since the implementation of the method only necessitates sharing information related to coupling variables, which connects agents through common constraints or objectives, and the method always outputs a feasible solution candidate at each iteration.

IPD was first formulated by (Zhao, 2001) for solving two stage linear stochastic programs and later applied to more generic convex stochastic programs (Mehrotra and Özçem, 2009b; Chen and Mehrotra, 2011). The method is mostly used in the stochastic programming context (Sarić et al., 2009) where the optimization is decomposed among scenarios that represent the possible realizations of an uncertain variable in the problem. Instead, we use IPD in the context of multi-agent optimization and decompose the problem among agents. The method is based on the primal barrier method (Bertsekas, 2008; Boyd and Vandenberghe, 2004) which converts a constrained optimization problem into an unconstrained one by using barrier functions.

Many decomposition based distributed optimization methods that are developed in the optimization research (Bertsekas and Tsitsiklis, 1997) are being extended and applied in multi-agent contexts (Necoara et al., 2011; Nedic and Ozdağlar, 2009; Nedic et al., 2010). Decomposition methods can be collected under two main categories: primal and

---

* This work has received support from the European Research Council under the European Unions Seventh Framework Programme (FP/2007-2013), ERC Grant Agreement 307608.
dual decomposition. In both approaches the optimization problem is split into a master problem that manages coupling data and many sub-problems that manage local data, although the master problem can often also be distributed. Dual decomposition methods are based on Lagrangian relaxation and therefore cannot guarantee primal feasibility until convergence. Primal decomposition methods on the other hand decompose the problem directly. In the presence of local constraints, the master problem becomes non-differentiable and is usually solved with sub-gradient or cutting plane methods (Bertsekas, 2008). These methods are able to preserve primal feasibility through all iterations and are therefore suitable for early termination, although convergence can be very slow (Bertsekas, 2008).

IPD on the other hand, is a second order primal decomposition method, as it solves unconstrained, or equality constrained, differentiable master and sub-problems with Newton’s method, therefore achieves fast convergence and primal feasibility.

Recently, several second order distributed optimization methods based on Newton’s method have been proposed (Pakazad et al., 2015; Necora and Suykens, 2009; Wei and Ozdaglar, 2012). These methods are similar to IPD as they apply the barrier method to their problem and compute Newton steps in a distributed fashion. The method in (Pakazad et al., 2015) is applicable to problems where agents are ‘loosely’ coupled and the descent direction is computed in a distributed manner without any central agent. On the other hand, IPD can handle more generic coupling constraints and cost functions by relying on a central agent. Similarly, the method proposed in (Wei and Ozdaglar, 2012) is applicable to a specific structure arising from network utility maximization. In (Necora and Suykens, 2009) the method is based on dual decomposition, and therefore cannot maintain primal feasible iterations.

IPD is essentially a decomposition of the primal barrier method. The problem is smoothed by using local and global barrier functions, allowing Newton’s method to be applied on the master problem, achieving fast convergence as well as feasible iterations. In the following sections we will briefly introduce the method, and discuss its attractive properties for solving a multi-agent optimization problem: fast convergence, primal feasible iterates for both local and coupling constraints, data hiding for local agents and problem size reduction for the central agent. For finding a feasible initial point and feasibility assessment in line-search, we propose novel methods based on simple surrogate sets (Bitlislioglu et al., 2017; Zhen and den Hertog, 2015). We also present an application of the method to an example from smart-grids: voltage support in (Wei and Ozdaglar, 2012). These methods are similar to the primal method (Bertsekas, 2008) and represents the optimal value of the problem (2) for a given $y_i$ that parameterizes the constraints of (2). The primal function $p_i(y_i)$ of a convex program is convex, and non-differentiable in general (Boyd and Vandenberghe, 2004).

2. PRELIMINARIES

2.1 Multi-agent Problem

We formulate the multi-agent problem with $n_a$ agents as

$$\begin{align*}
\min_{x,y} & \quad f_0(x,y) + \sum_{i=1}^{n_a} p_i(y_i) \\
\text{s. t.} & \quad Tx + Vy = r \\
& \quad h_j(x,y) \leq 0 \quad j \in [1, \ldots, l]
\end{align*}$$

(1)

where $x \in \mathbb{R}^{n_x}$ is a global variable, $y_i \in \mathbb{R}^{n_{y_i}}$ represents local outputs that introduce coupling between agents via the inequality constraint, the constraint functions $h_j$’s and the cost function $f_0$, which are all assumed to be convex. We consider the case where agents have additional internal variables $z_i$ and constraints affecting the output $y_i$. The local cost functions $p_i$’s are defined as

$$p_i(y_i) = \min_{z_i} f_i(z_i)$$

(2)

s. t. $P_i z_i = y_i$

$g_{i,j}(z_i) \leq 0, \quad j \in [1, \ldots, m_i]$

where $f_i$ and the $g_{i,j}$’s are convex functions and the matrix $P_i \in \mathbb{R}^{n_{y_i} \times n_{z_i}}$ has full row rank. $p_i(y_i)$ is called the ‘primal’ function (Bertsekas, 2008) and represents the optimal value of the problem (2) for a given $y_i$ that parameterizes the constraints of (2). The primal function $p_i(y_i)$ of a convex program is convex, and non-differentiable in general (Boyd and Vandenberghe, 2004).

2.2 Barrier method

Consider the following convex optimization problem:

$$\begin{align*}
\min_{x} & \quad f(x) \\
\text{s. t.} & \quad h_j(x) \leq 0, \quad j \in [1, \ldots, l] \\
& \quad Tx = r
\end{align*}$$

Instead of solving the inequality constrained problem above, we instead formulate a barrier augmented version with only equality constraints

$$\begin{align*}
\min_{x} & \quad f(x) + \beta \phi(x) : Tx = r \\
\end{align*}$$

(3)

where $\phi(.)$ is a strictly convex and self-concordant barrier function (Boyd and Vandenberghe, 2004) for the feasible set defined by constraints $h_j(x) \leq 0, \quad j \in [1, \ldots, l]$ and $\beta > 0$ is the barrier weighting parameter. A common choice for $\phi(.)$ is the logarithmic barrier defined as

$$\phi(x) = -\sum_{j=1}^{l} \ln(-h_j(x)).$$

In the standard barrier method, the equality constrained augmented problem (3) is solved with Newton’s method for a decreasing sequence $\{\beta_k\}$, with $\beta_k \to 0$. 

Notation: $\mathbb{R}^n$ denotes the Euclidean space of dimension $n$. If a variable is subscripted, as $y_i \in \mathbb{R}^{m_i}$, for $i \in [1, \ldots, n]$, we define the stacked vector $y \in \mathbb{R}^n$ as $y := [y_1^T, \ldots, y_n^T]^T$. Similarly if a variable is indexed as $y(k) \in \mathbb{R}^{n_k}$ for $k \in [1, \ldots, N]$, $y \in \mathbb{R}^n$ represents $[y(1)^T, \ldots, y(N)^T]^T$. Subscript indexing is used to specify local variables among agents, whereas parenthesis indexing is used to specify time steps in the MPC formulation. For a function with two arguments, as in $f(x,y)$ with $f : \mathbb{R}^{n_x \times n_y} \to \mathbb{R}$, $\nabla_x f(\bar{x}, \bar{y})$ represents the gradient vector and $\nabla_{xx} f(\bar{x}, \bar{y})$ represents the Hessian matrix with respect to the variable in the subscript, evaluated at $(\bar{x}, \bar{y})$. For a function with a single argument, we omit the subscript.
We now consider application of the barrier method to problem (1). The barrier augmented multi-agent problem can be written as
\[
\begin{align*}
\min_{x,y} \quad & f_0(x,y) + \beta \phi_0(x,y) + \sum_{i=1}^{n_a} \rho_i(\beta, y_i) \\
\text{s.t.} \quad & Tx + Vy = r
\end{align*}
\]
where the barrier augmented primal functions \( \rho_i(\beta, y_i) \) are defined as
\[
\rho_i(\beta, y_i) := \min_{z_i} \quad f_i(z_i) + \beta \phi_i(z_i) \\
\text{s.t.} \quad P_i z_i = y_i
\]
Here, \( \phi_i \) is the barrier function for the feasible set defined by the local constraints:
\[
\mathcal{G}_i = \{ z_i : g_{i,j}(z_i) \leq 0, \quad j \in [1, \ldots, m_i] \}.
\]
It was shown by (Chen and Mehrotra, 2011) that, when logarithmic barrier functions are used, \( \rho_i(\beta, \cdot) \) is a self-concordant barrier function family for the feasible set of the output variable \( y_i \), defined as
\[
\mathcal{F}_i := \{ y_i | \exists z_i, y_i = P_i z_i, z_i \in \mathcal{G}_i \}.
\]
For \( \rho_i \), an explicit description is not available, however at a given point \( y_i \), the value of the function as well as its gradient and Hessian can be obtained by solving the unconstrained convex optimization problem (5). The gradient of the Hessian of \( \rho \) is given by
\[
\begin{align*}
\nabla y \rho(\beta, y) &= -\lambda^* \\
\nabla^2 y \rho(\beta, y) &= \left( P (\nabla^2 z L(z^*, \lambda^*)^{-1} P^T \right)^{-1}
\end{align*}
\]
where \( L(z, \lambda) = f(z) + \beta \phi(z) + \lambda^T (Pz - y) \) and the optimal primal dual pair \((z^*, \lambda^*)\) satisfy the second order sufficient conditions for optimality (Bertsekas, 2008). The relations (7) can be derived by applying the implicit function theorem to the optimality conditions, see (Chen and Mehrotra, 2011).

3. INTERIOR POINT DECOMPOSITION METHOD

The interior point decomposition (IPD) method consists of applying the primal barrier method for solving the multi-agent problem (1). The primal-barrier method is described by Algorithm 1. To simplify notation we define the barrier-augmented multi agent cost function as
\[
\hat{f}(\beta, x, y) = f_0(x,y) + \beta \phi_0(x,y) + \sum_{i=1}^{n_a} \rho_i(\beta, y_i)
\]
The unconstrained minimization step in the primal-barrier method is carried out with Newton’s method, which requires the gradient and Hessian of the cost function, and iteratively minimizes a second order approximation of the barrier-augmented cost function.

Algorithm 2 Decomposed Newton’s method

3.1 Initialization with surrogate sets

For start-up, IPD requires an initial point that is feasible for both local and coupling constraints. If such a point is not available, one can solve a ‘Phase 1’ problem for finding one; for various options see (Boyd and Vandenberghe, 2004). Another possibility, also suggested by (Mehrotra and Özewnętr, 2009b), is to add slack variables with large penalties to the problem. Due to the distributed nature of IPD, these methods will require additional communication and optimization steps.

We propose using simple inner approximations of the feasible sets \( \mathcal{F}_i \)’s of local outputs. Note that an explicit description of \( \mathcal{F}_i \) is in general not available, even to the local agent itself. However, one can use the methods proposed in (Bitlislioglu et al., 2017; Zhen and den Hertog, 2015) to recover a simple surrogate set
\[
S_i \subset \mathcal{F}_i,
\]
with a relatively large volume (Zhen and den Hertog, 2015), using robust programming. For initialization, each
agent computes the set $S_t$, which can be an ellipsoidal or a polytopic set, and passes it to the central agent. The central-agent then solves a surrogate feasibility problem to find a feasible point that satisfies coupling constraints within the aggregate set. Feasibility problem can be formulated in different ways. One option is to minimize the maximum constraint violation within the set $S_t \times \cdots \times S_n$. If the minimizer is not feasible, the procedure can be repeated by re-computing the surrogate sets around the new point. This method is likely to generate a feasible point with very few communication steps between central and local agents, if the volumes of surrogate sets are substantial. In our application example in Section 4, a single step of the method was sufficient to find a strictly feasible point.

3.2 Line search

One of the most important ingredients of Newton’s method is the line-search, for determining the step-size in the descent direction. The step-size should be selected as to preserve feasibility and to guarantee a sufficient decrease at each iteration (Bertsekas, 2008). We can split the line-search into two steps; first, a feasibility search for determining the maximum step size that can maintain feasibility and second, a minimizing search for finding the step-size with a sufficient decrease in the cost function.

**Feasibility search with Dikin’s ellipsoid:** For a given descent direction, each agent needs to solve an additional feasibility problem in order to determine the maximum feasible step-size. However this can be avoided by limiting the search within a surrogate set that is communicated to the central agent together with gradient and Hessian information. The surrogate sets can be computed as mentioned in the previous section. However, the computational cost of finding $S_t$ at every step might be prohibitive depending on the dimensionality. For this reason we propose using a feasible ellipsoid that can be constructed using the Hessian of the barrier function which is already at hand.

**Lemma 1.** At any strictly feasible internal variable $\tilde{z}$ the ellipsoid defined by

$$E(\tilde{z}) = \{y : (y - P\tilde{z})^T (P(\nabla^2 f_\tilde{y}(\tilde{z}))^{-1}P)^{-1}(y - P\tilde{z}) < 1\}$$

is contained in the feasible set for the output variable $y$; $E(\tilde{z}) \subset F$.

**Proof.** At any strictly feasible $\tilde{z} \in G$, the Dikin’s ellipsoid is defined by

$$D(\tilde{z}) := \{z : (z - \tilde{z})^T \nabla^2 \phi(\tilde{z})(z - \tilde{z}) < 1\}.$$  

(12)

For a self concordant barrier function $\phi$ of set $G$, $D(\tilde{z})$ is contained within the set $G$; $D \subset G$ (Nesterov and Nemirovskii, 1994). It can be shown that $E(\tilde{z})$ is the image of the Dikin’s ellipsoid $D(\tilde{z})$, under the linear operator $P$. Therefore, $E(\tilde{z})$ is contained in the feasible set $F$ for the output variable $y$, which is the image of the set $G$ under $P$.

After solving the local problem (5) the agents can easily construct the ellipsoid $E(z^*)$ and pass it’s parameters to the central agent, avoiding any further communication for a feasibility check. The resulting step-size can be significantly smaller compared to an approach that uses the actual maximum step-size, however the burden of extra communication and optimization by all agents is eliminated.

**Minimizing search:** Minimizing line search is carried out until a sufficient descend condition (Boyd and Vandenberghe, 2004) is satisfied. For $0 < \alpha < 0.5$ the condition is given by

$$f(x + td) < f(x) + \alpha \nabla f(x)^Td.$$  

(13)

The standard method for finding a point satisfying (13) is backtracking, which consists of decreasing the step-size by a predetermined reduction factor $0 < \beta < 1$ until (13) is satisfied. This method will require communicating the step-size to individual agents, solution of the problems (5) and re-communication of the value to the central-agent. Depending on whether the burden of communication or optimization dominate, one can take different approaches, such as: aggressive backtracking, adaptive step-size that relates to the Newton decrement (Zhao, 2001; Mehrrot and Özveyn, 2009a; Wei and Özdaglar, 2012) or using pre-computed values collected from agents.

4. APPLICATION - VOLTAGE SUPPORT WITH COMMERCIAL BUILDINGS

In this section we present an application of IDP to a distributed MPC problem related to smart grids. We consider a case where a distribution grid operator coordinates a group of commercial buildings that joins a Demand Response (DR) program, in order to balance the voltage surge caused by a large solar plant.

The grid under consideration is a simplified version of the IEEE 123 test feeder, taken from (Bolognani and Zampieri, 2016), which is modeled as a single-phase system, and consists of 56 nodes. The power flow equations are linearized with the method of (Bolognani and Zampieri, 2016) that is suitable for distribution grids.

The grid is populated by a mix of commercial buildings, the properties of which are given below.

<table>
<thead>
<tr>
<th>Type</th>
<th>Cooling Cap.[kW]</th>
<th>Thermal Zones</th>
<th># in DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large Office</td>
<td>250</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>Small Office</td>
<td>30</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Warehouse</td>
<td>150</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

The data for the building models are obtained from (US Department of Energy: Energy Efficiency and Renewable Energy) and (OpenEI, 2016). We consider a summer scenario, where the cooling systems of buildings that participate in DR are controlled by MPC, which relies on a thermal model that is generated with the OpenBuild Toolbox (Gorecki et al., 2015). The grid is mostly populated by small offices representing static loads, whereas 12 buildings: two large offices, three warehouses and seven small offices contribute to the DR program for voltage support.

The grid operator is responsible for maintaining voltage magnitudes of all nodes within the $\pm 5\%$ of the nominal value. Starting from an initial grid state, it predicts the solar power in-feed and power injections from all nodes and solves the following multi-agent optimization problem.
tracking for sufficient decrease search are used, resulting in
shown in Figure 2. For the line-search, Dikin’s ellipsoids
feasible point, and maintains local and global feasibility as
agents and minimizes the maximum voltage magnitude
where the central agent collects feasible ellipsoids from
initialization the method explained in Section 3.1 is used,
half-hour, resulting in a horizon length
the time of day between 8:00 and 15:00, sampled every
Figure 1 shows the voltage profiles of all nodes without
and communicate with the grid operator that collects data
augmented cost function related to its power consumption
building, which is limited by actuation constraints. The
cost function represents the economic cost of electricity
cal power consumption
constrained to lie inside the comfort limits. The electri-
coupled to the thermal dynamics. The input to thermal dynamics
energy over the sampling period. The
means zone temperatures, which are
constrained to lie inside the comfort limits. The electrical
power consumption $y_i$ is modeled as a time varying
linear function of the thermal cooling power input to the
building, which is limited by actuation constraints. The
cost function represents the economic cost of electricity
determined by the price $\pi$ and penalizes deviation from the
reference temperature values that represent ideal comfort
conditions.

The problem (14) can be formulated as the standard multi-
agent problem (1) and thus can be solved with IPD.
In this setting, each building can compute the barrier
augmented cost function related to its power consumption
and communicate with the grid operator that collects data
from all-agents and implements the primal barrier method,
given in Algorithm 1, with Newton’s method, given in
Algorithm 2.

The results of the optimization are shown in Figures 1-4.
Figure 1 shows the voltage profiles of all nodes without
and with coordination. The control problem is solved for the
time of day between 8:00 and 15:00, sampled every
half-hour, resulting in a horizon length $N = 14$. For
initialization the method explained in Section 3.1 is used,
where the central agent collects feasible ellipsoids from
agents and minimizes the maximum voltage magnitude
among all nodes. After this step, IPD proceeds from a
feasible point, and maintains local and global feasibility as
shown in Figure 2. For the line-search, Dikin’s ellipsoids
around the current point for feasibility and regular back-
tracking for sufficient decrease search are used, resulting in

\[
\begin{align*}
\min_{y_i} & \sum_{i=1}^{n_a} p_i(y_i) \\
\text{s. t.} & v(k) = \sum_{i=1}^{n_a} G_i y_i(k) + v(k) \quad \forall k \in [1, \ldots, N], \\
& 0.95 \leq v(k) \leq 1.05 \quad \forall k \in [1, \ldots, N],
\end{align*}
\]

where $N$ is the time horizon, $v(k) \in \mathbb{R}^I$ represents the
predicted voltage magnitudes, in p.u., of all nodes at step $k$, $y_i \in \mathbb{R}^{n_{a_i}}$ represents the active power injections from
the buildings that participate in DR. $G_i \in \mathbb{R}^{I \times n_{a_i}}$ models
the effect of the injections from the $i$th node on the voltage
magnitudes of all nodes. We assume that the power factor
of the buildings are constant, and therefore reactive power
computations can be embedded inside $G_i$. The voltage
constraints are respected while the power injections are
allocated fairly according to the local cost functions $p_i$, defined as

\[
p_i(y_i) = \min_{u_i} \sum_{k=1}^{N} \|\theta_i(k) - \theta_{i,ref}\|^2_Q + \pi(k) y_i(k)
\]

\[
\begin{align*}
&\forall k \in [1, \ldots, N], \\
x_i(k+1) = A_i x_i(k) + B_i u_i(k) + E_i d_i(k) \\
\theta_i(k+1) = C_i x_i(k) + D_i u_i(k) \\
y_i(k) = F_i(k) u_i(k) \\
\theta_i^{min} \leq \theta_i(k) \leq \theta_i^{max}, \quad u_i^{min} \leq u_i(k) \leq u_i^{max}
\end{align*}
\]

Fig. 1. Predicted voltage magnitudes, without the demand
response coordination (top), and with optimal coordi-
nation of 12 buildings in the grid (bottom).

Fig. 2. The top figure shows the voltage magnitude at node
32, where the solar power infeed is placed. The figures
below show the zone temperatures and total cooling
power for a large office building placed at node 31.
a reasonable number of optimization and communication
steps, as shown in Figure 3.

Finally, we provide a comparison of IPD with a first
order distributed method: ADMM. In order to preserve
the same communication structure, we introduce global
copies of local outputs as $y_{glob} = y$ and apply ADMM to
the partial Lagrangian with respect to this constraint. By

Fig. 4. Comparison between ADMM and IPD applied to
the problem (14). The deviation from optimal profiles
is defined as the sum of absolute difference in MWh,
with respect to the optimal power consumption profiles,
for all buildings: $\sum_{i=1}^{n_a} \sum_{k=1}^{N_k} |y_i(k) - y_i(k^*)|$, with $\delta t$ being the sampling time of 30 minutes.

Fig. 3. Evolution of suboptimality measure $\|y - y^*\|_2$ for
the problem (14), using the Dikin’s ellipsoid feasibility
search and backtracking with a reduction factor of 0.8.
Total Newton steps are 58 whereas 77 communication steps were carried out. Barrier shrinking factor $\mu$ is set to 0.4.

doing this, one can split the primal updates into a central and
parallel local problems similar to the IPD method.
The penalty parameter for ADMM is adaptive as described in (Boyd, 2010). We can observe from Figure 4 that
ADMM quickly reduces the deviation from optimal energy
profiles, however the solution candidates remain infeasible
for many iterations. On the other hand IPD always takes
feasible steps by keeping the maximum voltage values
below limits and achieves much higher accuracy in a
smaller number of communication steps, compared to
ADMM.

Fig. 4. Comparison between ADMM and IPD applied to
the problem (14). The deviation from optimal energy
profiles, however the solution candidates remain infeasible
for many iterations. On the other hand IPD always takes
feasible steps by keeping the maximum voltage values
below limits and achieves much higher accuracy in a
smaller number of communication steps, compared to
ADMM.

REFERENCES

Belmont, Massachusetts.

Computation: Numerical Methods. Athena Scientific, Belmont,
Massachusetts.

Commitment. IEEE Transactions on Automatic Control,
PP(99), 1–1.

Approximation of the Power Flow Solution in Power Distribution

Boyd, S. (2010). Distributed Optimization and Statistical Learning via
the Alternating Direction Method of Multipliers. Foundations
and Trends in Machine Learning, 3(1), 1–122.


Decomposition-based Interior Point Methods for the Two-stage


Interior Point Decomposition Algorithms for Two-Stage Stochastic
Conic Programs. SIAM Journal on Optimization, 19(4), 1846–
1880.

Interior Point Methods for Two-Stage Stochastic Convex Quadratic

Necoara, I., Nedevschi, V., and Domínguez, I. (2011). Parallel and
distributed optimization methods for estimation and control in

Decomposition Method for Separable Convex Optimization. Journal

for Multi-Agent Optimization. IEEE Transactions on Automatic
Control, 54(1), 48–61.


algorithms in convex programming. SIAM, Philadelphia.


Primal-dual Interior-point Methods for Solving Loosely Coupled

stage stochastic programming model for market clearing with
contingencies. IEEE Transactions on Power Systems, 24(3),
1266–1278.

US Department of Energy: Office of Energy Efficiency and Renew-

Wei, E. and Ozdaglar, A. (2012). Distributed Alternating Direction Method of Multipliers. In 51st IEEE Conference on Decision and
Control (CDC), 5445–5450.

Zhao, G. (2001). A Log-Barrier method with Benders Decomposition
for solving two-stage stochastic linear programs. Mathematical
Programming, 90(3), 507–536.