DISTRIBUTED OPTIMIZATION VIA ADAPTIVE REGULARIZATION FOR LARGE PROBLEMS WITH SEPARABLE CONSTRAINTS

Elad Gilboa, Phani Chavali[*]*, Peng Yang*[*] *and Arye Nehorai*

{gilboae,chavalis, yangp, nehorai}@ese.wustl.edu
Preston M. Green Department of Electrical and Systems Engineering
Washington University in St. Louis, St. Louis, MO 63130

ABSTRACT

Many practical applications require solving an optimization over large and high-dimensional data sets, which makes these problems hard to solve and prohibitively time consuming. In this paper, we propose a parallel distributed algorithm that uses an adaptive regularizer (PDAR) to solve a joint optimization problem with separable constraints. The regularizer is adaptive and depends on the step size between iterations and the iteration number. We show theoretical convergence of our algorithm to an optimal solution, and use a multi-agent three-bin resource allocation example to illustrate the effectiveness of the proposed algorithm. Numerical simulations show that our algorithm converges to the same optimal solution as other distributed methods, with significantly reduced computational time.

1. INTRODUCTION

With the sensor and the storage technologies becoming increasingly cheaper, modern applications are seeing a sharp increase in *big data*. The explosion of such high-dimensional and complex data sets makes optimization problems extremely hard and prohibitively time consuming [1]. Parallel computing has received a significant attention lately as an effective tool to achieve the high throughput processing speeds required for processing big data sets. Thus, there has a been a paradigm shift from aggregating multi-core processors to utilizing them efficiently [2].

Although distributed optimization has been an increasingly important topic, it has not received sufficient attention since the seminal work by Bertsekas and Tsitsiklis until recently. In the 1980's, Bertsekas and Tsitsiklis extensively studied decentralized detection and consensus problems [3] and developed algorithms such as parallel coordinate descent [4] and the block coordinate descent (BCD) (also called the block Jacobi) [3, 5]. In 1994, Ferris *et. al.* proposed parallel variable distribution (PVD) [6] that alternates between a parallelization and a synchronization step. In the parallelization step, several sub-optimal points are found using parallel optimizations. Then, in the synchronization step, the optimal point is computed by taking an optimal weighted average of the points found in the parallel step. Although PVD claims to achieve better convergence rate than BCD, the complexity of solving optimization in both the steps make it impractical for high dimensional problems. There are other efficient distributive methods in literature, such as the shooting [7], the shotgun [2], and the alternating direction method of multipliers (ADMM) [1], however, these methods apply to only a specific type of optimization problems: ℓ_1-regularization for shooting and shotgun, and linear constraints for ADMM.

In this paper, we propose a fully distributed parallel method to solve optimization problems over high-dimensional data sets, which we call the parallel distributive adaptive regularization (PDAR). Our method can be applied to a wide variety of nonlinear problems where the constraints are block separable. The assumption of block separable constraints is valid for many practical problems, such as, multi-agent resource allocation where resources are being distributed amongst several agents that influence the choice of allocation. In order to coordinate among the subproblems we introduce an adaptive regularizer term that penalizes the large changes in successive iterations. Our method can be seen as an extension of the classical proximal point method (PPM) [8] with two novel advances. First, our motivation for using the PPM framework is very different than the original. We use PPM as a means to coordinate among the parallel subproblems and not for handling non differentiability. Second, we enforce coordination by using adaptive regularizers that vary across different subproblems.

The rest of the paper is organized as follows. In Section 2, we formulate the problem; in Section 3 we propose our parallel distributive algorithm and show convergence to an optimum solution; in Section 4 we provide numerical simulations, and we conclude the paper in Section 5.

2. PROBLEM FORMULATION

Consider an optimization problem given as:

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) && (1)\\ &\text{subject to} && \boldsymbol{x} \in \mathcal{X}, && (2) \end{aligned}$$

This work was supported by the Department of Defense under the AFOSR Grant FA9550-11-1-0210, and by National Science Foundation under the NSF Grants CCF-1014908 and CCF-0963742.

[*] Equally contributing authors

where the objective is to find the optimal vector $\boldsymbol{x}^*$ that minimizes the function $f(\boldsymbol{x}) \in \mathrm{R}$, with $\boldsymbol{x} \in \mathrm{R}^d$. The problem is often very complex, nonlinear, and high dimensional, and solving it is prohibitively time consuming. We assume that the constraint $\boldsymbol{x} \in \mathcal{X}$ can be separated into several blocks, such that

$$\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_i, \dots, \boldsymbol{x}_N] \text{ where, } \boldsymbol{x}_i \in \mathcal{X}_i, \tag{3}$$

with $\boldsymbol{x}_i \in \mathrm{R}^{n_i}$ and $\sum_{i=1}^N n_i = d$. Once the problem is separated into blocks, distributed iterative approaches (such as the ones mentioned in the Introduction section) can be applied. However, these methods are time consuming when the subproblems are themselves complex.

3. DISTRIBUTED OPTIMIZATION VIA ADAPTIVE REGULARIZATION

In this section, we describe our distributed optimization framework with adaptive regularization. We solve the optimization problem given by Eq. (1) in a parallel and iterative manner. Let k denote the iteration index and $\hat{\boldsymbol{x}}^k = (\hat{\boldsymbol{x}}_i^k, \hat{\boldsymbol{x}}_{-i}^k)$, with $\hat{\boldsymbol{x}}_{-i}^k = \left[\hat{\boldsymbol{x}}_1^k, \dots, \hat{\boldsymbol{x}}_{i-1}^k, \hat{\boldsymbol{x}}_{i+1}^k, \dots, \hat{\boldsymbol{x}}_N^k\right]$ denote the solution to the optimization problem in the k^{th} iteration. In order to obtain a solution in a distributed manner, we define a set of N augmented objective functions at each iteration k as [1]

$$L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1}) = f(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_{-i}^{k-1}) + \lambda_i^k(\boldsymbol{h}_i^{k-1})\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i^{k-1}\|^2, \tag{4}$$

where $\boldsymbol{h}_i^{k-1} = \hat{\boldsymbol{x}}_i^{k-1} - \hat{\boldsymbol{x}}_i^{k-2}$ is the step taken by the i^{th} block in the $(k-1)^{th}$ iteration, and $\lambda_i^k(\boldsymbol{h}_i^{k-1})$ is an adaptive regularization coefficient which depends on both the indices i and k. We will describe the form of this regularization coefficient shortly. After defining the objective functions $L_i^k(\cdot),\ i = 1, \dots, N$, we solve N optimization problems in a parallel fashion:

$$\begin{aligned}
\hat{\boldsymbol{x}}_1^k &= \arg\min_{\boldsymbol{x}_1 \in \mathcal{X}_1} L_i^k(\boldsymbol{x}_1; \hat{\boldsymbol{x}}^{k-1}), \\
\hat{\boldsymbol{x}}_2^k &= \arg\min_{\boldsymbol{x}_2 \in \mathcal{X}_2} L_i^k(\boldsymbol{x}_2; \hat{\boldsymbol{x}}^{k-1}), \\
&\vdots \\
\hat{\boldsymbol{x}}_N^k &= \arg\min_{\boldsymbol{x}_N \in \mathcal{X}_N} L_i^k(\boldsymbol{x}_N; \hat{\boldsymbol{x}}^{k-1}).
\end{aligned} \tag{5}$$

This optimization framework is in the form of a decomposition-coordination procedure [1], where N agents are trying to minimize their own augmented objective functions, and the new joint vector $\hat{\boldsymbol{x}}^k$ is obtained by simply aggregating the N blocks. If we consider a single objective function $L_i^k(\boldsymbol{x}_i^k)$ at a single iteration k, the minimization of the objective functions is only with respect to he variables of the i^{th} block. However, since the objective function depends also on variables from other blocks, a change in them will cause a change to the objective function, namely $L_i^{k+1}(\boldsymbol{x}_i^k) \neq L_i^k(\boldsymbol{x}_i^k)$.

[1]We use a semicolon notation in Eq. (4) to clarify that only the variables on the left of the semicolon are allowed to change.

```
Algorithm: PDAR
k = 1; % Iteration counter
Initialize $\boldsymbol{x}^0$ and $\lambda_i^0\ \forall\ i$
do
    parfor $i$ in $1:N$
        $\hat{\boldsymbol{x}}_i^k = \arg\min_{\boldsymbol{x}_i \in \mathcal{X}_i} L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1})$
        Set $\boldsymbol{h}_i^k = \hat{\boldsymbol{x}}_i^k - \hat{\boldsymbol{x}}_i^{k-1}$
        Update $\lambda_i^k$
    end parfor
    $k := k + 1$
until $\|f(\boldsymbol{x}^k) - f(\boldsymbol{x}^{k-1})\| \leq \delta$
```

Table 1: Algorithm for Parallel Distributed Optimization

Next, we discuss the choice of the regularization coefficient $\lambda_i^k(\boldsymbol{h}_i^{k-1})$. We chose $\lambda_i^k(\boldsymbol{h}_i^{k-1})$ to be of the form:

$$\lambda_i^k(\boldsymbol{h}_i^{k-1}) = \begin{cases} \max(\phi(\|\boldsymbol{h}_i^{k-1}\|), \beta) & \text{if } k < K \\ \alpha k & \text{otherwise,} \end{cases} \tag{6}$$

where K is a threshold on the iteration index, $\alpha > 0$, and $\beta > 0$ are parameters chosen depending on the problem. Intuitively, the threshold K divides each optimization problem into two phases. The goal of the first phase is to coordinate the parallel optimization. In this phase, each of the agents change their solution in response to the solutions of other agents. This alternating behavior can be enforced by choosing the function $\phi(\|\boldsymbol{h}_i^{k-1}\|)$ to be a nondecreasing with respect to $\|\boldsymbol{h}_i^{k-1}\|$. This choice will increase the value of regularization coefficient, $\lambda_i^k(\boldsymbol{h}_i^{k-1})$ as $\|\boldsymbol{h}_i^{k-1}\|$ increases. The increase in $\lambda_i^k(\boldsymbol{h}_i^{k-1})$ will in turn enforce a smaller stepsize on the agents that had large change in the previous iteration, to allow other agents to react in the current iteration. The goal of the second phase is to fine tune the solution and to enable it to reach a local optimum. In this paper we choose $\phi(\|\boldsymbol{h}_i^{k-1}\|) = N^2\|\boldsymbol{h}_i^{k-1}\|$. The algorithm is summarized in Table 1.

3.1. Discussion on the Convergence

In this section, we show that the algorithm described in the previous subsection converges to an optimum solution. Assume that the function $f(\boldsymbol{x})$ is convex. Since the augmented function $L_i^k(\cdot)$, $i = 1, \dots, N$ is the sum of two convex functions, it is convex. We then have

$$\hat{\boldsymbol{x}}_i^k = \arg\min_{\boldsymbol{x}_i \in \mathcal{X}_i} L_i^k(\boldsymbol{x}_i; \hat{\boldsymbol{x}}^{k-1}). \tag{7}$$

Since $\hat{\boldsymbol{x}}_i^k$ is a minimizer of $L_i^k(\boldsymbol{x}_i;\hat{\boldsymbol{x}}^{k-1})$, we have by the first order necessary conditions for local optimum that

$$\begin{aligned}
&\boldsymbol{\nabla}_i L_i^k(\boldsymbol{x}_i;\hat{\boldsymbol{x}}^{k-1})\Big|_{\boldsymbol{x}_i=\hat{\boldsymbol{x}}_i^k} = 0,\\
&\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k,\hat{\boldsymbol{x}}_{-i}^{k-1}) + 2\lambda_i^k(\boldsymbol{h}_i^{k-1})\underbrace{(\hat{\boldsymbol{x}}_i^k-\hat{\boldsymbol{x}}_i^{k-1})}_{\boldsymbol{h}_i^k} = 0,\\
\Rightarrow\quad &\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k,\hat{\boldsymbol{x}}_{-i}^{k-1}) = -2\lambda_i^k(\boldsymbol{h}_i^{k-1})\boldsymbol{h}_i^k,\\
\Rightarrow\quad &\boldsymbol{h}_i^k = \frac{-\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k,\hat{\boldsymbol{x}}_{-i}^{k-1})}{2\lambda_i^k(\boldsymbol{h}_i^{k-1})}, \qquad (8)
\end{aligned}$$

where the operator $\boldsymbol{\nabla}_i$ is a gradient operator with respect to $\boldsymbol{x}_i$. For $k > K$ we have $\lambda_i^k(\boldsymbol{h}_i^{k-1}) = \alpha k$, and therefore Eq. (8) simplifies as

$$\boldsymbol{h}_i^k = \frac{1}{2\alpha k}\underbrace{\Big(-\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k,\hat{\boldsymbol{x}}_{-i}^{k-1})\Big)}_{\boldsymbol{d}_i^k}, \qquad (9)$$

where $\boldsymbol{d}_i^k$ is the negative gradient direction of the i^{th} agent. By concatenating all the directions into a single vector $\boldsymbol{d}^k = [\boldsymbol{d}_1^k, \boldsymbol{d}_2^k, \dots, \boldsymbol{d}_N^k]$, we get the next iterate $\boldsymbol{x}^k$ as

$$\hat{\boldsymbol{x}}^k \;=\; \hat{\boldsymbol{x}}^{k-1} + \boldsymbol{h}^k, \qquad (10)$$

where $\boldsymbol{h}^k = \frac{\boldsymbol{d}^k}{2\alpha k}$. We prove the convergence properties of the algorithm using the following two prepositions.

Proposition 1: For the sequence of non-stationary iterates $\hat{\boldsymbol{x}}^k$ obtained from the PDAR algorithm, $\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{d}^k < 0$. [2]
Proof: From the definition of $\boldsymbol{d}_i^k$, we have

$$\boldsymbol{d}_i^k = -\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k,\hat{\boldsymbol{x}}_{-i}^{k-1}). \qquad (11)$$

Therefore,

$$\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{d}^k = \sum_{i=1}^N -\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}^{k-1})\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k,\hat{\boldsymbol{x}}_{-i}^{k-1}). \qquad (12)$$

Since $\hat{\boldsymbol{x}}_i^k$ is a result of minimizing $L_i^k(\boldsymbol{x}_i;\hat{\boldsymbol{x}}^{k-1})$, the corresponding step $\boldsymbol{h}_i^k$ must be in a descending direction. Thus

$$\boldsymbol{\nabla}_i L_i^k(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{h}_i^k = \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{h}_i^k \le 0,\ \forall\, i \qquad (13)$$

However, there must exist at least one block where the strict inequality $\boldsymbol{\nabla}_i f(\boldsymbol{x}_i^{k-1})'\boldsymbol{h}_i^k < 0$ holds. We prove this by contradiction. Assume that $\forall i,\ \boldsymbol{\nabla}_i f(\boldsymbol{x}_i^{k-1})'\boldsymbol{h}_i^k = 0$. If $\boldsymbol{h}_i^k = 0, \forall i$, then $\hat{\boldsymbol{x}}^k$ is a stationary point which contradicts the assumption of convergence to a nonstationary point. Hence there exists some i, for which $\boldsymbol{h}_i^k \neq 0$. Now, since $L_i^k(\boldsymbol{x}^k;\hat{\boldsymbol{x}}^{k-1})$ is a convex function, it must lie above all of its tangents, i.e.,

$$L_i^k(\hat{\boldsymbol{x}}_i^k;\hat{\boldsymbol{x}}_{-i}^{k-1}) \ge L_i^k(\hat{\boldsymbol{x}}^{k-1}) + \boldsymbol{\nabla}_i L_i^k(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{h}_i^k. \qquad (14)$$

Since $\boldsymbol{\nabla}_i L_i^k(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{h}_i^k = \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{h}_i^k = 0$, we have from Eq. (14), that $L_i^k(\hat{\boldsymbol{x}}^k) \ge L_i^k(\hat{\boldsymbol{x}}^{k-1})$. This is a contradiction, since every iterate should reduce the objective function corresponding to the block. Intuitively, this inequality implies that if the step size is perpendicular to the gradient of the objective function, then such steps do not decrease the value of the objective function. Hence there exists at least one block that satisfies inequality $\boldsymbol{\nabla}_i f(\boldsymbol{x}_i^{k-1})'\boldsymbol{h}_i^k < 0$. Finally, since at least one block satisfies the strict inequality, their summation satisfies strict inequality:

$$\begin{aligned}
&\sum_{i=1}^N \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^{k-1})'\boldsymbol{h}_i^k < 0,\\
\Rightarrow\quad &\sum_{i=1}^N \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k,\hat{\boldsymbol{x}}_{-i}^{k-1}) < 0,\\
\Rightarrow\quad &\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{d}^k < 0.
\end{aligned}$$

Proposition 2: Assume that f gradients to be uniformly continuous in the ℓ_2 norm, and that its gradients are bounded. The sequence $\hat{\boldsymbol{x}}^k$ converges to an optimal solution.
Proof: Formally, we need to show that for any subsequence $\{\hat{\boldsymbol{x}}^k\}$ that converges to a nonstationary point, the corresponding subsequence $\{\boldsymbol{d}^k\}$ is bounded and satisfies [8]:

$$\lim_{k\to\infty}\ \mathrm{sup}_{k\in\mathcal{K}}\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{d}(\hat{\boldsymbol{x}}^k) < 0, \qquad (15)$$

where $\boldsymbol{d}(\hat{\boldsymbol{x}}^k) = -\sum_{i=1}^N \boldsymbol{\nabla}_i f(\hat{\boldsymbol{x}}_i^k,\hat{\boldsymbol{x}}_{-i}^{k-1})$. Let $\epsilon > 0$, and $\{\boldsymbol{x}^k\}_{k\in\mathcal{K}}$ be an arbitrary sequence of nonstationary points such that

$$\lim_{k\to\infty}\ \mathrm{sup}_{k\in\mathcal{K}}\hat{\boldsymbol{x}}^k = \bar{\boldsymbol{x}},$$

where $\boldsymbol{\nabla} f(\bar{\boldsymbol{x}}) \neq 0$. Then $\forall k \in \mathcal{K}$ the gradients are not equal to zero, $\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^k) \neq 0$, since the sequence has nonstationary points. Using Proposition 1, we have that $\forall k \in \mathcal{K}$, $\boldsymbol{\nabla} f(\hat{\boldsymbol{x}}^{k-1})'\boldsymbol{d}(\boldsymbol{x}^k) < 0$, and specifically $\boldsymbol{\nabla} f(\bar{\boldsymbol{x}})'\boldsymbol{d}(\bar{\boldsymbol{x}}) = D_1 < 0$.

By the continuity assumption of the gradients, there $\exists\,\delta > 0$ such that $\|\boldsymbol{\nabla} f(\boldsymbol{y})'\boldsymbol{d}(\boldsymbol{y}) - \boldsymbol{\nabla} f(\bar{\boldsymbol{x}})'\boldsymbol{d}(\bar{\boldsymbol{x}})\| < \epsilon, \forall\ \|\boldsymbol{y}-\bar{\boldsymbol{x}}\| < \delta$. Since $\boldsymbol{x}^k \to \bar{\boldsymbol{x}}$, $\exists\, N\ \in \mathrm{N}$ such that $\forall k > N$, $\|\boldsymbol{x}^k - \bar{\boldsymbol{x}}\| < \delta$, and thus

$$\|\boldsymbol{\nabla} f(\boldsymbol{x}^{k-1})'\boldsymbol{d}(\boldsymbol{x}^k) - \boldsymbol{\nabla} f(\bar{\boldsymbol{x}})'\boldsymbol{d}(\bar{\boldsymbol{x}})\| < \epsilon.$$

This implies that $\boldsymbol{\nabla} f(\boldsymbol{x}^{k-1})'\boldsymbol{d}(\boldsymbol{x}^k) < D_1 + \epsilon$. As $\epsilon > 0$ is arbitrary, $\lim_{k\to\infty}\mathrm{sup}_{k\in\mathcal{K}}\boldsymbol{\nabla} f(\boldsymbol{x}^{k-1})'\boldsymbol{d}(\boldsymbol{x}^k) = D_1 < 0$. Hence the sequence of iterates $\boldsymbol{x}^k$ converges to an optimal solution.

[2]For brevity, if all the blocks in the function are from the same iteration, we will simplify the notation, i.e., $f(\hat{\boldsymbol{x}}_i^{k-1},\hat{\boldsymbol{x}}_{-i}^{k-1}) = f(\hat{\boldsymbol{x}}^{k-1})$

4. NUMERICAL RESULTS

In this section, we provide numerical results to compare the convergence of the proposed distributed algorithm to those of the block coordinate descent (BCD) and parallel variable distribution (PVD) . We consider a three-bin resource allocation example for the numerical simulation. Let there be $N = 100$ agents. Each agent has fixed quantity of resources that are to be allocated among three bins. Let $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, x_{i,3}]'$ denote the allocation scheme of the i^{th} agent. Without loss of generality, let $\sum_{j=1}^{3} x_{i,j} = 1, \ \forall\, i$. The objective is to minimize the sum of the individual costs, where the cost of agents depends on their own scheme and the schemes of other agents.

Let $\boldsymbol{x} = [\boldsymbol{x}'_1, \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_N]'$ denote the collective scheme of all agents. The cost function of the i^{th} agent is taken as

$$f_i(\boldsymbol{x}) = \boldsymbol{x}'_i \boldsymbol{P}_i \boldsymbol{g}(\boldsymbol{x}), \tag{16}$$

where $\boldsymbol{P}_i = \mathrm{diag}(p_{i,1}, p_{i,2}, p_{i,2})$ denotes the preference matrix of the i^{th} agent for each bin, and $\boldsymbol{g}(\boldsymbol{x}) = [g_1, g_2, g_3]'$ is a function dependent on the schemes of all agents, with

$$g_m = \left(\sum_{i=1}^{N} x_{i,m} \right)^2, \quad m \in \{1, 2, 3\}. \tag{17}$$

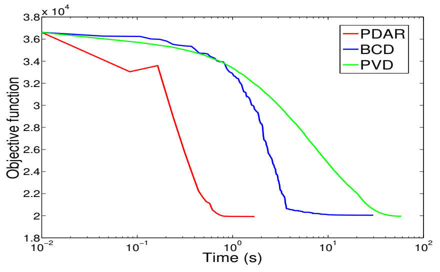
The goal is to solve the optimization problem:

$$\min_{\boldsymbol{x}} \sum_{i=1}^{N} f_i(\boldsymbol{x}) \ \text{ subject to } \sum_{j=1}^{3} x_{i,j} = 1, \forall\, i. \tag{18}$$

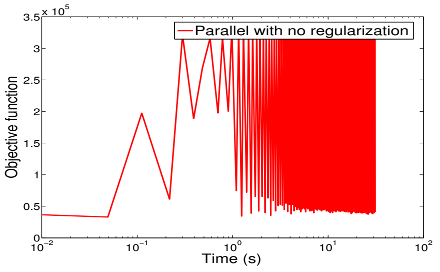
In order to find the solution to the above joint optimization problem, we solved $N = 100$ subproblems in parallel using our proposed PDAR. The optimization problem of the i^{th} agent in the k^{th} iteration is given as

$$\begin{aligned} \min_{\boldsymbol{x}_i} \quad & f_i(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_{-i}^{k-1}) + \lambda_i^k(\boldsymbol{h}_i^{k-1}) \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i^{k-1}\|^2 \\ \text{subject to} \quad & \sum_{j=1}^{3} x_{i,j} = 1. \end{aligned} \tag{19}$$

In Fig. 1a, we plot the value of the objective function as a function of the "normalized time" for BCD, PVD and our PDAR approach. We say "normalized time" to note the runtime of the parallel algorithms if we were not limited by the number of cores. In our example, we ran all the simulations on a 4 core machine; however in principle the parallel methods can run on 100 cores simultaneously. In order to make the comparison computer independent, the time axis corresponding to parallel methods was divided by 25. As illustrated, the convergence rate of our method is of an order of magnitude faster compared to BCD and PVD algorithms. The advantage comes from the fact that we can solve all the 100 optimization problems in parallel, whereas BCD is a sequential method. The PVD method, on the other hand, is worse even though it has a parallel update step. The additional time it takes to converge is due to the synchronization step, and due to the complexity of the optimization problems that are to be solved in both steps. In Fig. 1b, we show the oscillatory behavior when the parallel algorithm is used with out a regularizer. This figure further emphasizes the importance of a regularizer.



(a)



(b)

Fig. 1: Value of the objective functions vs time for the three bin resource allocation problem. Fig. 1a shows that PDAR converges much faster compared to BCD and PVD. Fig. 1b shows the oscillatory behavior of the parallel optimization without regularization.

5. CONCLUSIONS

In this paper, we proposed a distributed optimization framework to solve large optimization problems with separable constraints. Each agent solves a local optimization problem, which is much simpler compared to the joint optimization. In order for the agents to coordinate among themselves and to reach an optimum solution, we introduced a regularization term that penalized the changes in the successive iterations with an adaptive regularization coefficient. We proved that our solution always converges to a local optimum, and to a global optimum if the overall objective function is convex. Numerical simulations showed that the solutions reached by our algorithm are the same as the ones obtained using other distributed approaches, with significantly reduced computation time.

6. REFERENCES

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[2] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin, "Parallel coordinate descent for l1-regularized loss minimization," in *International Conference on Machine Learning (ICML 2011)*, Bellevue, Washington, June 2011.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[4] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. AC-31, no. 9, pp. 803–812, 1986.

[5] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, 2001.

[6] M. C. Ferris and O. L. Mangasarian, "Parallel variable distribution," *SIAM Journal on Optimization*, vol. 4, pp. 815–832, 1994.

[7] W. J. Fu, "Penalized regressions: The bridge versus the LASSO," *Journal of Comp. and Graphical Statistics*, vol. 7, no. 3, pp. 397–416, 1998.

[8] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.