Distributed Non-Convex ADMM-inference in Large-scale Random Fields

Ondrej Miksik[1]
http://www.miksik.co.uk
Vibhav Vineet[1]
vibhav.vineet@gmail.com
Patrick Pérez[2]
patrick.perez@technicolor.com
Philip H. S. Torr[1]
http://www.robots.ox.ac.uk/~tvg/

[1] Department of Engineering Science
University of Oxford
Oxford, UK

[2] Technicolor Research & Innovation
Cesson Sévigné, FR

Abstract

We propose a parallel and distributed algorithm for solving discrete labeling problems in large scale random fields. Our approach is motivated by the following observations: i) very large scale image and video processing problems, such as labeling dozens of million pixels with thousands of labels, are routinely faced in many application domains; ii) the computational complexity of the current state-of-the-art inference algorithms makes them impractical to solve such large scale problems; iii) modern parallel and distributed systems provide high computation power at low cost. At the core of our algorithm is a tree-based decomposition of the original optimization problem which is solved using a non convex form of the method of alternating direction method of multipliers (ADMM). This allows efficient parallel solving of resulting sub-problems. We evaluate the efficiency and accuracy offered by our algorithm on several benchmark low-level vision problems, on both CPU and Nvidia GPU. We consistently achieve a factor of speed-up compared to dual decomposition (DD) approach and other ADMM-based approaches.

1 Introduction

Probabilistic graphical models such as the Markov Random Fields (MRF) and Conditional Random Fields (CRF) [8], and related energy-minimization based techniques have become ubiquitous in computer vision and image processing. They have been proven especially useful to solve a variety of important, high-dimensional, discrete inference problems. Examples include per-pixel object labelling, image denoising, image inpainting, disparity and optical flow estimation, etc. [15, 18, 24]. Their use nonetheless implies computational costs that are often not compatible with very large scale problems met today in many applications. This concern is at the heart of present work.

As a result of more than two decades of impressive progress, a number of sophisticated algorithms have been proposed to solve, exactly or approximately, the combinatorial optimization problems associated to such approaches. However, their computation and/or memory complexity depends often dramatically on the size of search space (*e.g.*, the number of pixels raised to the power of the number of discrete labels) and on the structure of the energy (*e.g.*, the number of neighbours per pixel). As a consequence, algorithms that work well on smaller benchmarks can become impractical on very large scale problems, such as labeling millions of pixels with thousands of labels.

While recent advances in combinatorial optimization for computer vision have focused on important guarantees of convergence, this is not sufficient to achieve desired efficiency on large scale problems. In particular, given limited number of cpu cores, speed limitations of hard-drives and high costs of shared memory systems, massively parallel processors present an appealing computing paradigm. Thus, it becomes of paramount importance that new optimization algorithms for random fields can run in a parallel and distributed fashion on platforms such as Amazon EC2 [1] or Nvidia GPUs [12].

Parallelization has been considered for belief propagation [14], graph cuts [17, 20, 21], linear programming and dual decomposition [7]. Graph-cut is generally solved by augmenting-path or push-relabel algorithms. The former are generally unsuited for parallel processing [4]. Push-relabel algorithms involve more local operations, but it has been shown [20] that their parallel implementation generally oscillates between two solutions and takes numerous iterations to converge. Loopy belief propagation [25], on the other hand, involves local updates, but does not come with convergence guarantees in general.

The approaches based on linear programming (LP) relaxation appear to be, by construction, the best suited to parallel and distributed treatment. One class of such approaches, the dual decomposition [7], solves the Lagrangean relaxation of the integer linear programming (ILP) problem and decomposes the large and difficult problem to solve into a set of smaller sub-problems that can be easily solved in parallel, with minimal communication through a master problem. Solution of the latter combines the solutions of the sub-problems in a principled manner and ensures consistency/agreement among them. The dual decomposition approach has very strong theoretical properties, however it suffers from certain issues in practice. When using sub-gradient optimization [7], there is no guarantee of monotonic improvement, hence the convergence rate is very slow; when using block coordinate ascent instead [16, 19], convergence to the dual optimum is not guaranteed. Finally, proximal methods that can solve the relaxed LP-problem in primal do not allow closed form updates in general [13].

An alternative way to decompose the original problem is proposed by the Alternating Direction Method of Multipliers (ADMM) method [3]. ADMM provides a robust convex optimization approach with stronger and better properties than the dual decomposition approach, *e.g.* guarantees global convergence at $O(1/\varepsilon)$ rate (ε is error) even when the functions are non-smooth [23]. In recent years, ADMM has been successfully used to solve large scale relaxed LP problems in a distributed fashion. Examples include AD3 [9], ADLP [11] or Bethe-ADMM [5]. For instance, the AD3 approach is effective when it decomposes the problem into very small size subproblems, typically involving a pair of neighboring variables only. On large scale problems, this unfortunately leads to a very large number of such subproblems and to a rather slow convergence.

In this work, we propose a highly efficient, fully distributed algorithm for large scale discrete inference problems, which is also based on ADMM. Normally ADMM requires decomposition into convex functions. Instead we decompose into non-convex functions, but which can still be solved exactly because they are submodular. The decomposition is done into tree-based sub-problems of arbitrary sizes, each one being solved through dynamic programming. Whilst exact solving of each sub-problem does not guarantee convergence to the global optimum empirically good results are found. One of our key contributions is to demonstrate that the memory and computation complexities of the approach are only of the order of the size of sub-problems. Our approach is easy to implement, since each sub-problem requires one call to a dynamic programming solver, and is highly suitable for modern GPUs with thousands of CUDA cores. Finally, we show empirically that our approach rapidly converges to a good quality estimates and is able to return a solution at any point in practice, which is important when developing interactive systems.

We extensively evaluate the efficiency and accuracy of our distributed inference algorithm on two platforms: on a CPU and on Nvidia GPU. We show convergence and speed on different benchmark problems such as disparity estimation and image segmentation.

2 Setting the Stage

2.1 Integer Linear Program Formulation

We first define a discrete random field $\mathbf{Y} = \{y_1, y_2, \dots, y_N\}$ attached to the N nodes of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. Each random variable takes a label from a discrete space $\mathcal{L}$ of size L. We define $\mathcal{Y} = \mathcal{L}^N$ the set of all possible label assignments. This random field is a pairwise Markov Random Field (MRF) if there exists an energy function of the form

$$E(\mathbf{Y}) := \sum_{i \in \mathcal{V}} \theta_i(y_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(y_i, y_j), \tag{1}$$

composed of unary and pairwise potentials. Finding the lowest cost labeling of the energy over $\mathcal{Y}$ is an NP-hard combinatorial problem which can be written as the Integer Linear Program (ILP) [22]

$$\begin{aligned} \mathbf{ILP-MRF}: \ \text{minimize} \ & \sum_{i \in \mathcal{V}} \boldsymbol{\theta}_i \cdot \boldsymbol{p}_i + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\theta}_{ij} \cdot \boldsymbol{q}_{ij} \\ \text{with respect to} \ & (\boldsymbol{p}, \boldsymbol{q}) \in \text{Marg}(\mathcal{G}). \end{aligned} \tag{2}$$

Here $\boldsymbol{\theta}_i = \{\theta_i(l)\}_{l \in \mathcal{L}} \in \mathbb{R}^L$ represents a vector of unary potentials for i-th variable taking a label $l \in \mathcal{L}$ and $\boldsymbol{\theta}_{ij} = \{\theta_{ij}(l, l')\}_{l, l' \in \mathcal{L}}$ corresponds to a vector of pairwise potentials for a pair of variables (y_i, y_j) taking labels l and l' of size $L \times L$. We define $\boldsymbol{p} = \{\boldsymbol{p}_i\}_{i \in \mathcal{V}}$ and $\boldsymbol{q} = \{\boldsymbol{q}_{ij}\}_{(i,j) \in \mathcal{E}}$ as vectors of binary indicators consisting of all unary $\boldsymbol{p}_i = \{p_i(.)\}$ and pairwise $\boldsymbol{q}_{ij} = \{q_{ij}(.,.)\}$ subvectors. The indicator $p_i(l)$ represents the assignment of label l to the variable y_i, *i.e.* $p_i(l) = 1$ iff label l is assigned to the variable y_i and zero otherwise. Similarly $q_{ij}(l, l') = 1$ iff label l is assigned to variable y_i and l' to y_j. These indicator vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ are of length $P = L \times N$ and $Q = L \times L \times |\mathcal{E}|$ respectively. Since all the possible assignments $(\boldsymbol{p}, \boldsymbol{q})$ take only integral solutions, they are constrained to lie in the marginal polytope $\text{Marg}(\mathcal{G})$ [22] which is defined as:

$$\text{Marg}(\mathcal{G}) = \left\{ \begin{array}{l} \boldsymbol{p} = \{\boldsymbol{p}_i\}_{i \in \mathcal{V}} \ \ \in \{0,1\}^P \\ \boldsymbol{q} = \{\boldsymbol{q}_{ij}\}_{ij \in \mathcal{V}} \in \{0,1\}^Q \end{array} \middle| \begin{array}{ll} \sum_{l \in \mathcal{L}} p_i(l) = 1, & \forall i \in \mathcal{V} \\ \sum_{l' \in \mathcal{L}} q_{ij}(l, l') = p_i(l) & \forall (i,j) \in \mathcal{E}, l \in \mathcal{L} \\ \boldsymbol{p}_i \in \{0,1\}^L & \forall i \in \mathcal{V} \\ \boldsymbol{q}_{ij} \in \{0,1\}^{L \times L} & \forall (i,j) \in \mathcal{E} \end{array} \right\}. \tag{3}$$

The integral constraints $\boldsymbol{p}_i \in \{0,1\}^L$ and $\boldsymbol{q}_{ij} \in \{0,1\}^{L \times L}$ lead to a non-convex minimization problem (ILP-MRF problem in Eq. 2). A pair $(\boldsymbol{p}, \boldsymbol{q}) \in \text{Marg}(\mathcal{G})$ satisfies the marginal constraints as given in Eq. 3 (non-negativity, summing-up to one, and marginal consistency between a pairwise-vector and the pixel-wise vectors on corresponding nodes).

The ILP-MRF is in general NP-hard problem and many methods have been proposed to solve it approximately. One such class of methods depends on solving the LP-relaxation of the original integer problem defined in Eq. 2. More importantly our focus is on algorithms which solve the LP-problem in distributed fashion, typically based on block coordinate descent [6] or on projected sub-gradient method [7]. Block coordinate descent is usually faster and enjoys optimality guarantees if the relaxation is tight [16], but may get stuck at sub-optimal solutions when the objective is non-smooth. In contrast, projected sub-gradient based method can provably solve the dual problem, however it is not a descend method (no guarantee of monotonic improvement) requiring a sequence of diminishing step-sizes and hence its convergence is slow [7]. In order to overcome this issue we design an ADMM-based inference algorithm to solve the original ILP-MRF problem.

2.2 Alternating Direction Method of Multipliers

We now briefly describe general Alternating Direction Method of Multipliers (ADMM) approaches for solving optimization problems, then we describe a special case of ADMM – a convex *consensus* problem and finally we discuss non-convex problems. This will form our basis for solving the ILP-MRF problem (Eq. 2) which we outline in Sec. 3.

ADMM is a powerful optimization technique that combines the benefits of dual decomposition and method of multipliers. The basic idea revolves around decomposing a large (and probably difficult) problem into a set of (simpler to solve) subproblems and cleverly combining their solutions in a principled manner to recover the solution of the original problem. Such ADMM based decomposition strategies have been recently applied to efficiently solve MAP estimation problem [9, 10]. To describe the ADMM approach, we first consider a general convex optimization problem

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \end{aligned} \tag{4}$$

where both functions $f(\mathbf{x})$ and $g(\mathbf{z})$ are convex, closed and proper. The objective function is separable across variables $\mathbf{x}$ and $\mathbf{z}$, but the problem itself is not decomposable because of the equality constraint $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$. This problem can be turned into an unconstrained minimization problem by introducing the *augmented* Largrangian:

$$L_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) := f(\mathbf{x}) + g(\mathbf{z}) + \boldsymbol{\lambda} \cdot (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \underbrace{\frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2}_{\text{quadratic penalty}}, \tag{5}$$

where $\boldsymbol{\lambda}$ is the dual variable as in classic Lagrangian duality and ρ is a positive parameter. While the additional penalty destroys the separability as compared to classic Lagrangian, it helps solving dual problem efficiently. The ADMM approach conducts the joint optimization of augmented Lagrangian by alternating the following three steps:

$$\mathbf{x}^{(t+1)} := \arg\min_{\mathbf{x}} \left[f(\mathbf{x}) + \boldsymbol{\lambda}^{(t)} \cdot (\mathbf{A}\mathbf{x}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^{(t)} - \mathbf{c}\|_2^2 \right], \tag{6}$$

$$\mathbf{z}^{(t+1)} := \arg\min_{\mathbf{z}} \left[g(\mathbf{z}) + \boldsymbol{\lambda}^{(t)} \cdot (\mathbf{B}\mathbf{z}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x}^{(t+1)} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2 \right], \tag{7}$$

$$\boldsymbol{\lambda}^{(t+1)} := \boldsymbol{\lambda}^{(t)} + \rho \left(\mathbf{A}\mathbf{x}^{(t+1)} + \mathbf{B}\mathbf{z}^{(t+1)} - \mathbf{c} \right). \tag{8}$$

First, augmented Lagrangian is minimized over $\mathbf{x}$, keeping the other variables fixed (Eq. 6), and likewise over second variable $\mathbf{z}$ (Eq. 7). Freezing primal variables, dual variables

$\boldsymbol{\lambda}$s are then updated (Eq. 8) using one step of gradient ascent with step-size equal to the augmented Lagrangian parameter ρ. Updating $\mathbf{x}$ and $\mathbf{z}$ in turn rather than jointly allows for decomposition. It requires that these alternate updates can be computed efficiently if not in a closed form.

The ADMM approach can be used to solve convex *consensus* optimization problem. For this, consider the following optimization problem

$$\text{minimize} \qquad f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}), \tag{9}$$

where each f_i is convex, closed and proper. The problem is not separable because the variable $\mathbf{x}$ is shared by all the subproblems. In order to make the problem separable, we first introduce a set of local variables $\mathbf{x}_i$ and a global variable $\mathbf{z}$ and rewrite Eq. 9 as the *consensus* problem:

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{n} f_i(\mathbf{x}_i) \\ &\text{subject to} && \mathbf{x}_i - \mathbf{z} = 0, \quad i = 1, \ldots, n. \end{aligned} \tag{10}$$

Even though the objective function is separable across local variables $\mathbf{x}_i$, the problem is still not decomposable because of the consensus constraints $\mathbf{x_i} = \mathbf{z}, \forall i$. As in the case of general ADMM approach, we next convert this into an unconstrained maximization problem by introducing the *augmented* Lagrangian:

$$L_\rho(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n) = \sum_{i=1}^{n} \left(f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i \cdot (\mathbf{x}_i - \mathbf{z}) + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}\|_2^2 \right). \tag{11}$$

The ADMM updates for the *consensus* problem are:

$$\mathbf{x}_i^{(t+1)} := \arg\min_{\mathbf{x}_i} \left[f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^{(t)} \cdot (\mathbf{x}_i - \mathbf{z}^{(t)}) + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}^{(t)}\|_2^2 \right], \tag{12}$$

$$\mathbf{z}^{(t+1)} := \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}_i^{(t+1)} + \frac{1}{\rho} \boldsymbol{\lambda}_i^{(t)} \right), \tag{13}$$

$$\boldsymbol{\lambda}_i^{(t+1)} := \boldsymbol{\lambda}_i^{(t)} + \rho \left(\mathbf{x}_i^{(t+1)} - \mathbf{z}^{(t+1)} \right). \tag{14}$$

The first and last steps can now be carried out independently for each subproblem and the second step exchanges messages only with the relevant subproblems, *i.e.* in a decentralized manner. First, the augmented Lagrangian is minimized over each $\mathbf{x}_i$ independently, keeping the other variables fixed (Eq. 12), followed by the update for the global variable $\mathbf{z}$ (Eq. 13). Finally each dual variable $\boldsymbol{\lambda}_i$ is then updated (Eq. 14) independently using one step of gradient ascent with step-size equal to the augmented Lagrangian parameter ρ.

Next we briefly review the application of ADMM to distributed non-convex optimization. Let us again take our optimization problem from Eq. 9 but now consider

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{n} f_i(\mathbf{x}) \\ &\text{subject to} && \mathbf{x} \in \mathcal{C}, \end{aligned} \tag{15}$$

where $\mathcal{C}$ is a non-convex set. ADMM solving a problem with non-convex constraints modifies the second step of the above problem as:

$$\mathbf{z}^{(t+1)} := \mathcal{P}_{\mathcal{C}} \left(\frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}_i^{(t+1)} + \frac{1}{\rho} \boldsymbol{\lambda}_i^{(t)} \right) \right), \tag{16}$$

where $\mathcal{P}_{\mathcal{C}}$ is a projection onto a non-convex set $\mathcal{C}$. The update steps for $\mathbf{x}_i$ and $\boldsymbol{\lambda}_i$ are the same as for the *consensus* optimization problem described earlier. However, the $\mathbf{z}$-update is now a projection onto a non-convex set. In general, it is hard to compute but it can be carried out easily and exactly in special cases. The most relevant case is when the elements in $\mathcal{C}$ are only binary variables which is of interest to us. For example, if $\mathcal{C} = \{\mathbf{x} | x_i \in \{0,1\}\}$, then the projection step $\mathcal{P}_{\mathcal{C}}$ simply rounds each entry to 0 or 1, whichever is closer. More discussions can be found in the paper of Boyd *et al.* [3].

3 Distributed Optimization

3.1 ADMM-based Decomposition

Following [7], we split the original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into S sub-graphs $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, $s = 1 \ldots S$ and associate to each one auxiliary variables $\boldsymbol{p}^s = \{\boldsymbol{p}_i^s\}_{i \in \mathcal{V}_s}$ and $\boldsymbol{q}^s = \{\boldsymbol{q}_{ij}^s\}_{(i,j) \in \mathcal{E}_s}$, and potential parameters $\{\boldsymbol{\theta}_i^s,\ i \in \mathcal{V}_s\}$ and $\{\boldsymbol{\theta}_{ij}^s,\ (i,j) \in \mathcal{E}_s\}$, such that:

$$\sum_{s: i \in \mathcal{V}_s} \boldsymbol{\theta}_i^s = \boldsymbol{\theta}_i,\ \forall i \in \mathcal{V}; \qquad \sum_{s:(i,j) \in \mathcal{E}_s} \boldsymbol{\theta}_{ij}^s = \boldsymbol{\theta}_{ij},\ \forall (i,j) \in \mathcal{E}. \tag{17}$$

This implies that each node and each edge of the original graph must be covered by at least one sub-graph and that the sub-graphs can share freely nodes and edges and that the potentials on all shared vertices or edges of the sub-graphs sum to that of the original graph.

Given sub-graphs and associated parameters, we aim to replace the difficult inference problem (2) by a set of sub-problems that can be solved in parallel, while consistency between them is enforced in some way. Within the ADMM framework, there are several ways to achieve this goal. We choose to rely on "master" variables $\boldsymbol{p} = \{\boldsymbol{p}_i\}_{i \in \mathcal{V}}$ at the node level only. Thanks to constraints (17), it is easy to see that the original ILP-MRF problem can be written as

$$\begin{aligned} \mathbf{DIP-MRF}: \text{ minimize } \quad & \sum_{s=1}^{S} \left(\sum_{i \in \mathcal{V}_s} \boldsymbol{\theta}_i^s \cdot \boldsymbol{p}_i^s + \sum_{(i,j) \in \mathcal{E}_s} \boldsymbol{\theta}_{ij}^s \cdot \boldsymbol{q}_{ij}^s \right) \\ \text{with respect to } \quad & (\boldsymbol{p}^s, \boldsymbol{q}^s) \in \mathrm{Marg}(\mathcal{G}_s),\ \forall s \\ & \boldsymbol{p}_i \in \{0,1\}^L,\ \forall i \in \mathcal{V} \\ \text{subject to } \quad & \boldsymbol{p}^s = \boldsymbol{p}_{|s},\ \forall s \end{aligned} \tag{18}$$

where $\boldsymbol{p}_{|s} = \{\boldsymbol{p}_i\}_{i \in \mathcal{V}_s}$ denotes the sub-vector of $\boldsymbol{p}$ containing variables only for nodes of $s-$th sub-graph.

The master node variables $\boldsymbol{p}_i$'s do not appear in the new objective function. Consensus between them and the node variables of all sub-graphs is enforced via equality constraints. Note that this consensus, along with the consistency that marginal polytope enforces between node and edge variables of each sub-graph, also ensures consensus between edge variables across overlapping sub-graphs.

Looking at review chapter in [3] we can see that the form of Eq. 18 follows a special case of optimization problems know as *consensus* for which there exists simplified forms for the ADMM update. Following this, we first form the augmented Lagrangian of problem

$$L_\rho(\{(\boldsymbol{p}^s, \boldsymbol{q}^s)\}, \boldsymbol{p}, \{\boldsymbol{\lambda}^s\}) = \sum_{s=1}^{S} \left(E_s(\boldsymbol{p}^s, \boldsymbol{q}^s; \boldsymbol{\theta}^s) + \sum_{i \in \mathcal{V}_s} \boldsymbol{\lambda}_i^s \cdot (\boldsymbol{p}_i^s - \boldsymbol{p}_i) + \frac{\rho}{2} \sum_{i \in \mathcal{V}_s} \|\boldsymbol{p}_i^s - \boldsymbol{p}_i\|_2^2 \right) \tag{19}$$

where $E_s(\boldsymbol{p}^s, \boldsymbol{q}^s; \boldsymbol{\theta}^s) = \sum_{i \in \mathcal{V}_s} \boldsymbol{\theta}_i^s \cdot \boldsymbol{p}_i^s + \sum_{(i,j) \in \mathcal{E}_s} \boldsymbol{\theta}_{ij}^s \cdot \boldsymbol{q}_{ij}^s$, $\boldsymbol{p} \in \{0,1\}^P$, $(\boldsymbol{p}^s, \boldsymbol{q}^s) \in \text{Marg}(\mathcal{G}_s)$ and $\boldsymbol{\lambda}^s = \{\boldsymbol{\lambda}_i^s\}_{i \in \mathcal{V}_s} \in \mathbb{R}^{L \times |\mathcal{V}_s|}$. This is a consensus problem in that we essentially have multiple copies of the same variable that should take the value of the master. We can derive the following much simplified update rules for ADMM [3].

Broadcast Involves solving one inference problem per sub-graph:

$$(\boldsymbol{p}^s, \boldsymbol{q}^s)^{(t+1)} := \underset{(\boldsymbol{p}^s, \boldsymbol{q}^s) \in \text{Marg}(\mathcal{G}_s)}{\text{argmin}} L_\rho(\{(\boldsymbol{p}^s, \boldsymbol{q}^s)\}, \boldsymbol{p}^{(t)}, \{\boldsymbol{\lambda}^{s(t)}\}), \quad \forall s. \tag{20}$$

Each of these inference problems is a quadratic program (QP). Normally solving this set of QP's would be prohibitively expensive, in computation and memory, however one of the key contributions of this paper is to show how we can do this efficiently in Section 3.2.

Gather In the gathering step, we need to minimize L_ρ w.r.t. $\boldsymbol{p}$ under the constraint that the $\boldsymbol{p}$ lies in the marginal polytope $\text{Marg}(\mathcal{G})$ which is a non-convex problem. So we follow the projection step described in Eq. 16 as

$$\boldsymbol{p}_i^{(t+1)} := \mathcal{P}_{\text{Marg}(\mathcal{G})} \left(\frac{1}{|\mathcal{I}(i)|} \sum_{s \in \mathcal{I}(i)} \left(\boldsymbol{p}_i^{s(t+1)} + \frac{1}{\rho} \boldsymbol{\lambda}_i^{s(t)} \right) \right), \qquad \forall i \in \mathcal{V}, \tag{21}$$

where $\mathcal{I}(i) = \{s : i \in \mathcal{V}_s\}$ and $\mathcal{P}_{\text{Marg}(\mathcal{G})}$ involves projection onto $\text{Marg}(\mathcal{G})$ which simply rounds the value to 0 or 1, whichever is closer. This step can be computed in a decentralized manner – each $\boldsymbol{p}_i$ is independent and needs an access only to the corresponding nodes (solutions) of sub-problems into which the original potential was split. This update gets simplified if ρ is fixed through iterations and $\boldsymbol{\lambda}^{s(t=0)} = \mathbf{0}$:

$$\boldsymbol{p}_i^{(t+1)} := \mathcal{P}_{\text{Marg}(\mathcal{G})} \left(\frac{1}{|\mathcal{I}(i)|} \sum_{s \in \mathcal{I}(i)} \boldsymbol{p}_i^{s(t+1)} \right), \qquad \forall i \in \mathcal{V}. \tag{22}$$

Multiplier update The dual variable update involves only a step of dual ascent with step-size ρ which can be made iteration dependent for improved convergence:

$$\boldsymbol{\lambda}_i^{s(t+1)} := \boldsymbol{\lambda}_i^{s(t)} + \rho \left(\boldsymbol{p}_i^{s(t+1)} - \boldsymbol{p}_i^{(t+1)} \right), \quad \forall s, \ \forall i \in \mathcal{V}_s. \tag{23}$$

3.2 QP Sub-problems

We now describe our key contribution to solve each subproblem efficiently. Since we have not relaxed the problem (Eq. 2), we can not use any general-purpose convex optimization solver or custom solvers such as AD3 [10], ADLP [11]. Further these methods generally do not scale to handle large scale random field problems consisting of millions of variables and hundreds of classes, making them impractical for computer vision problems. To this end, we present an alternative approach, which reduces the difficult quadratic problem (Eq. 20) to an equivalent simpler to solve problem. But this simplification leads to an algorithm that has similar computational and memory complexity as that of belief propagation.

In order to set up the problem, we again take the QP problem corresponding to $s-$th sub-problem (Eq. 20) to be solved at iteration t:

$$\begin{aligned} \mathbf{QP-s:} \ \text{minimize} \ & E_s(\boldsymbol{p}^s, \boldsymbol{q}^s; \boldsymbol{\theta}^s) + \sum_{i \in \mathcal{V}_s} \boldsymbol{\lambda}_i^{s(t)} \cdot (\boldsymbol{p}_i^s - \boldsymbol{p}_i^{(t)}) + \frac{\rho}{2} \sum_{i \in \mathcal{V}_s} \|\boldsymbol{p}_i^s - \boldsymbol{p}_i^{(t)}\|_2^2 \\ \text{with respect to} \quad & (\boldsymbol{p}^s, \boldsymbol{q}^s) \in \text{Marg}(\mathcal{G}_s). \end{aligned} \tag{24}$$

Elementary manipulations of the objective, and removal of constant terms yields the equivalent objective

$$\begin{aligned} \mathbf{QP-s:}\ \text{minimize}\ & E_s(\boldsymbol{p}^s,\boldsymbol{q}^s;\boldsymbol{\theta}^s+\boldsymbol{\lambda}^{s(t)}-\rho\cdot\boldsymbol{p}^{(t)}_{|s})+\frac{\rho}{2}\|\boldsymbol{p}^s\|_2^2 \\ \text{with respect to}\quad & (\boldsymbol{p}^s,\boldsymbol{q}^s)\in\text{Marg}(\mathcal{G}_s). \end{aligned} \tag{25}$$

Next we describe our efficient algorithm solving the above problem.

Efficient inference For the next discussion, we constrain the sub-graphs to be trees or chains so that we can use belief propagation (BP) to help solving associated sub-problems.

Lemma 1. *Given that* $\boldsymbol{p}_i^s\in\text{Marg}(\mathcal{G}_s)$, *the binary indicator vector* $\boldsymbol{p}_i^s$ *in Eq. 25 is a unit vector with a single non-zero entry which is 1.*

Lemma 2. *The dot-product between two same unit vectors with a single non-zero entry is always equal to one, i.e.* $\boldsymbol{p}_i^s\cdot\boldsymbol{p}_i^s=1$.

Theorem 1. *The QP subproblems Eq. 24, 25 reduces to a standard belief propagation with adjusted singleton terms for tree-structured subproblems.*

$$\begin{aligned} (\boldsymbol{p}^s,\boldsymbol{q}^s)^{(t+1)} := \arg\min_{(\boldsymbol{p}^s,\boldsymbol{q}^s)}\quad & E_s(\boldsymbol{\theta}^s+\boldsymbol{\lambda}^{s(t)}-\rho\cdot\boldsymbol{p}^{(t)}_{|s};\boldsymbol{p}^s,\boldsymbol{q}^s)+\text{const.} \\ \text{s.t.}\quad & (\boldsymbol{p}^s,\boldsymbol{q}^s)\in\text{Marg}(\mathcal{G}_s). \end{aligned} \tag{26}$$

Proof. Under assumptions Lemma 1 and Lemma 2, we observe that the quadratic term in Eq. 25 is always constant as it is a dot product of two same unit vector with a single non-zero entry.

$$(\boldsymbol{p}^s,\boldsymbol{q}^s)^{(t+1)} := \operatorname*{arg\,min}_{(\boldsymbol{p}^s,\boldsymbol{q}^s)\in\text{Marg}(\mathcal{G}_s)} \left(E_s(\boldsymbol{\theta}^s+\boldsymbol{\lambda}^{s(t)}-\rho\cdot\boldsymbol{p}^{(t)}_{|s};\boldsymbol{p}^s,\boldsymbol{q}^s)+\frac{\rho}{2}\sum_{i\in\mathcal{V}_s}\underbrace{\boldsymbol{p}_i^s\cdot\boldsymbol{p}_i^s}_{=1}\right). \tag{27}$$

A term with a constant value for any possible label assignment does not effect the energy minimization. Each subproblem minimizes the following energy function

$$\begin{aligned} (\boldsymbol{p}^s,\boldsymbol{q}^s)^{(t+1)} := \arg\min_{(\boldsymbol{p}^s,\boldsymbol{q}^s)}\quad & E_s(\boldsymbol{\theta}^s+\boldsymbol{\lambda}^{s(t)}-\rho\cdot\boldsymbol{p}^{(t)}_{|s};\boldsymbol{p}^s,\boldsymbol{q}^s)+\text{const.} \\ \text{s.t.}\quad & (\boldsymbol{p}^s,\boldsymbol{q}^s)\in\text{Marg}(\mathcal{G}_s). \end{aligned} \tag{28}$$

We replaced the quadratic term by adjusted unary potentials. Since we do not need to introduce any loops the QP subproblem reduces to a standard belief propagation with adjusted singleton terms. ■

4 Experimental Results

We demonstrate the performance of our algorithm on several low-level benchmark labelling problems including stereo correspondence, image segmentation and denoising. In all experiments, timings are based on a single Nvidia GTX Titan Black graphics card with 6144 MB memory on board featuring 15 multiprocessors and 2880 stream processors. Our CPU timings are based on code run on an Intel Xeon 3.33 GHz CPU with 24 GB of RAM.

These low-level problems highlights different properties of our algorithm. The review paper of Szeliski *et al.* [18] provides many benchmark images corresponding to these problems. Further, they provided the unary and pairwise energy terms as part of their dataset for different benchmark problems which we use in all our experiments. Our main comparison on GPU is with the dual-decomposition (DD) based approach of Komodakis *et al.* [7]. To highlight the efficiency and accuracy achieved by our algorithm against other existing distributed algorithms such as the DD approach and ADMM based AD3 approach, we extensively evaluate different properties on a single core CPU machine. Finally we also compare against tree re-weighted message passing (TRWS) algorithm. We use the codes in the software library OpenGM for the DD and an implementation provided by authors of AD3.

In Fig. 1 we show the energy values and lower bounds (dual function) across different iterations of our algorithm on three benchmark images: Flower (image segmentation), Tsukuba (stereo correspondence) and Penguin (image denoising). We compare the energies and lower bounds achieved across different iterations of our algorithm against the other distributed algorithms and the TRW-S approach. It can seen that our algorithm achieves a solution very close to the optimum in a few iterations (generally 20 iterations), while the dual decomposition based method takes $3-5\times$ more iterations to converge to the same solution. Additionally our algorithm achieves global optimum for binary submodular problem as in binary image segmentation problem following weak tree agreement [7]. In general on all these benchmark images, our algorithm achieves better energies than the TRW-S approach except on image denoising problem where TRW-S performs better than us.

We also extensively compare the speed-up offered by our approach on GPU and multi-core CPUs. In Tab. 1 we compare the time taken by our algorithm against the other approaches to reach to the same level of accuracy on binary image segmentation problem (Flower). As can be seen, our algorithm takes almost $5.2\times$ less time than the approach of the dual decomposition method to reach to an error rate of 0.001% on Nvidia GPU. The CPU version is approximately $2\times$ faster than the AD3. The dual decomposition uses projective adaptive stepsize. For AD3, we used adaptive stepsize and caching. For both methods (DD and AD3), we run the algorithms for various initial step-sizes from range $< 0.01, 100 >$ and report the best results. Our method uses adaptive stepsize rule based on residuals, lazy evaluation (caching) and a medium-size subproblems. Next we provide details on the sizes of subproblems, and the idea of caching on image segmentation problem.

Subproblem size Even though in theory all the tree-structured subproblems are equally powerful, in practice, we observe that different sizes of tree-structured subproblems effect runtime differently. Figure 1 (right) demonstrates this behaviour – while the larger subproblems require less iterations to converge, the smaller subproblems converge in a shorter runtime up to some subproblem-size as they allow to cache larger part of the problem. As shown in Fig. 1 the number of iterations taken by sub-problems of sizes 2 is generally very large (around 1600) iterations compared to when we have each subproblem of size 200 which takes around 300 iterations to converge. Empirically we found that the tree of size 20 worked best for us by converging to the optimum solution fastest.

Caching/Lazy-evaluation After a few iterations, solutions $\boldsymbol{p}_i^s$ of many subproblems achieve a consensus, *i.e.* $\boldsymbol{p}_i^{s(t)} = \boldsymbol{p}_{|s,i}^{(t)}, \forall s \in S$ and do not change over iterations $\boldsymbol{p}_i^{s(t)} = \boldsymbol{p}_i^{s(t+1)}$. This means, that also the global variables $\boldsymbol{p}_i^{(t)}$ are the same and the dual variables $\boldsymbol{\lambda}_i^s$ are not changing any more. If all variables $\boldsymbol{p}_i^s$ of a subproblem s achieve a consensus, the subproblem s enters the idle state and does not need to be re-computed in the next iteration. Therefore our

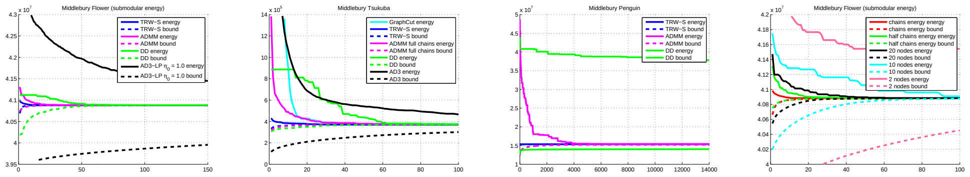

Figure 1: The energy values and lower bounds (dual function) across different iterations of our algorithm. **A and B:** Comparison against dual decomposition (DD) on the GPU for image segmentation ("Flower") and disparity estimation ("Tsukuba") problems. Our approach requires less iterations to reach to the same accuracy than both, DD and AD3. **C:** comparison of fixed-ρ ADMM and DD with square summable but not summable stepsize for image denoising ("Penguin"). **D:** We show how different sizes of chains effect the convergence.

Algorithm / ε	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
C-AD3 (CPU)	0.0400	30.0000	89.0000	189.8400	264.2000	288.2800	293.1000
TRWS (CPU)	0.1000	0.1100	0.4092	0.9736	2.3482	4.1452	5.7330
DD (CPU)	0.6500	1.0000	6.5000	24.8900	90.0800	322.6500	438.2500
DD (GPU)	0.0094	0.0185	0.0940	0.0357	1.416	4.7000	6.3450
ADMM (CPU)	0.7046	5.9971	23.1602	78.1000	97.8220	109.5700	134.8900
ADMM (GPU)	0.0688	0.0601	0.2316	0.7263	0.9202	1.0064	1.2661
C-ADMM (GPU)	**0.0688**	**0.0315**	**0.0568**	**0.1520**	**0.1952**	**0.2267**	**0.2854**

Table 1: Time taken (in seconds) by different algorithms on binary segmentation ("Flower") for both, CPU and GPU implementations. The timings are reported w.r.t. error ε from the bound, *i.e.* $\varepsilon = 10^{-1}$ is worse than $\varepsilon = 10^{-7}$. CPU implementations do not favour parallelism (evaluated on a single core). A GPU implementation with caching of the proposed method (C-ADMM) is: a) 480 $\times$ faster than the CPU version (without caching); b) 5.5$\times$ faster than the gpu implementation of the DD (both without caching); c) our gpu implementation is 1027$\times$ faster than the cpu version of AD3 (both with caching) and 20$\times$ faster than TRWS.

inference approach only put computation efforts where they are required. In Tab. 1 we show how GPU timings changes by a factor of 10$\times$ after adding caching into the computation.

We observe that algorithms with lazy evaluation are in particular useful when we have to do with a very large number of labels. Finally it should be noted that our focus was on speed (implementation). We believe with the advancement in the GPU and multi-core architectures, we will be able to perform experiments on even larger random fields with billions of variables.

5 Conclusion

In this work, we have proposed an efficient distributed inference algorithm for MAP estimation in random field models. It enjoys the same decompositional properties as the dual decomposition approach but in practice gives better performance. The proposed algorithm converges more quickly than its competitors and is suitable for a GPU implementation. We observe a significant improvement in speed compared to various state-of-the-art methods.

Our approach fits well with the recent interests in developing efficient distributed algorithms dealing with the large-scale problems. We believe there might be several future works spawning from our algorithm, *e.g.* distributed learning of large-scale random fields.

Acknowledgement: We would like to thank André Martins and Jörg Kappes for fruitful discussions. This research was supported by Technicolor, EPSRC, the Leverhulme Trust and the ERC grant ERC-2012-AdG 321162-HELIOS.

References

[1] Amazon. Amazon elastic compute cloud (amazon ec2). *https://aws.amazon.com/ec2/.*

[2] Dimitri P. Bertsekas. *NonLinear Programming*. 2005.

[3] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9), 2004.

[5] Qiang Fu, Huahua Wang, and Arindam Banerjee. Bethe-ADMM for tree decomposition based parallel map inference. *CoRR*, abs/1309.6829, 2013.

[6] Amir Globerson and Tommi Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *NIPS*, 2007.

[7] Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(3): 531–552, 2011.

[8] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[9] André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. An augmented lagrangian approach to constrained map inference. In *ICML*, pages 169–176, 2011.

[10] André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. Alternating directions dual decomposition. *CoRR*, abs/1212.6550, 2012.

[11] Ofer Meshi and Amir Globerson. An alternating direction method for dual map lp relaxation. In *ECML/PKDD (2)*, pages 470–483, 2011.

[12] Nvidia. Tesla gpu accelerators for servers. *http://www.nvidia.co.uk/object/tesla-server-gpus-uk.html.*

[13] Pradeep D. Ravikumar, Alekh Agarwal, and Martin J. Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *Journal of Machine Learning Research*, 11:1043–1080, 2010.

[14] Alexander G. Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Distributed message passing for large scale graphical models. In *CVPR*, pages 1833–1840, 2011.

[15] Jamie Shotton, John M. Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1), 2009.

[16] David Sontag and Tommi Jaakkola. Tree block coordinate descent for map in graphical models. In *AISTATS*, pages 544–551, 2009.

[17] Petter Strandmark and Fredrik Kahl. Parallel and distributed graph cuts by dual decomposition. In *CVPR*, pages 2085–2092, 2010.

[18] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappena, and C. Rother. A comparative study of energy minimization methods for markov random fields. *TPAMI*, 30(6), 2007.

[19] Daniel Tarlow, Dhruv Batra, Pushmeet Kohli, and Vladimir Kolmogorov. Dynamic tree block coordinate ascent. In *ICML*, pages 113–120, 2011.

[20] Vibhav Vineet and P. J. Narayanan. Cuda cuts: Fast graph cuts on the gpu. In *CVPRW (3)*, pages 1–8, 2008.

[21] Vibhav Vineet and P. J. Narayanan. Solving multilabel MRFs using incremental *alpha*-expansion on the GPUs. In *ACCV (3)*, pages 633–643, 2009.

[22] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[23] Huahua Wang and Arindam Banerjee. Online alternating direction method. In *ICML*, 2012.

[24] Oliver J. Woodford, Philip H. S. Torr, Ian D. Reid, and Andrew W. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *CVPR*, 2008.

[25] J. Yedidia, W. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, 2000.