Greedy Direction Method of Multiplier for MAP Inference of Large Output Domain

Xiangru Huang †
Qixing Huang †
Ruohan Zhang †
Pradeep Ravikumar ‡
Inderjit S. Dhillon †
† University of Texas at Austin
‡ Carnegie Mellon University

Abstract

Maximum-a-Posteriori (MAP) inference lies at the heart of Graphical Models and Structured Prediction. Despite the intractability of exact MAP inference, approximate methods based on LP relaxations have exhibited superior performance across a wide range of applications. Yet for problems involving large output domains (i.e., the state space for each variable is large), standard LP relaxations can easily give rise to a large number of variables and constraints which are beyond the limit of existing optimization algorithms. In this paper, we introduce an effective MAP inference method for problems with large output domains. The method builds upon alternating minimization of an Augmented Lagrangian that exploits the sparsity of messages through greedy optimization techniques. A key feature of our greedy approach is to introduce variables in an on-demand manner with a pre-built data structure over local factors. This results in a single-loop algorithm of sublinear cost per iteration and $O(\log(1/\epsilon))$-type iteration complexity to achieve $\epsilon$ sub-optimality. In addition, we introduce a variant of GDMM for binary MAP inference problems with a large number of factors. Empirically, the proposed algorithms demonstrate orders of magnitude speedup over state-of-the-art MAP inference techniques on MAP inference problems including Segmentation, Protein Folding, Graph Matching, and Multilabel prediction with pairwise interaction.

1 Introduction

Graphical Models and Structured Prediction have become prevalent in a wide range of applications including Computer Vision, Natural Language Processing, Social Networks, and Computational Biology. A common theme in these problems is that the elements in output space are highly dependent. Such dependencies are parametrized by factors in the language of graphical models, and the prediction of structured outputs can be formulated as a problem of Maximum-a-Posteriori (MAP) inference, which finds the mode of the output distribution — that is, an output configuration maximizing the summation of factor values.

Although MAP inference problems are NP-hard in general, approximation methods based on various convex relaxations have proven to be effective, e.g., Linear Programming (LP) [1], Quadratic Programming (QP) [2] and Semidefinite Programming (SDP) [3]. Among them the LP-based relaxation exhibits the best trade-off between accuracy and efficiency (particularly on large-scale problems) and is arguably the most investigated approach in recent years [1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14].

In the context of linear programming relaxations for MAP inference, efficient algorithms usually take advantage of the underlying graph structures. However, the number of variables in an LP relaxation still grows linearly with both the number of factors and the size of factor domains, where the latter can be quadratic or even cubic in the number of states of each random variable. Therefore, for problems with large output domains, most existing linear programming techniques become intractably slow. This paper addresses the following question: can we iterate over only a small portion of potential labels for each random variable and still ensure a desired convergence rate to an optimal solution?

The contribution of this research is two-folded. First, we show that a greedy version of the Augmented La-
grangian Method which only passes a sparse set of active messages can possess a \(O(\log(1/\epsilon))\)-type iteration complexity. In particular, we propose two variants of the greedy schemes—Frank-Wolfe (FW) and Block-Greedy Coordinate Descent (BGCD)—for problems with a large state space and problems with a large number of factors, respectively. Second, we show by maintaining a sparse set of messages, the generation of active states or active factors can be performed in time sublinear to the number of variables utilizing a pre-built data structure. Therefore the proposed algorithm is orders-of-magnitude faster than state-of-the-art MAP inference algorithms on several problems with large output domains, including Protein Folding, Graph Matching, and Pairwise-Interacted Multilabel Prediction.

2 Related Work

The optimization procedures of the existing LP relaxations for MAP inference share a similar underlying mechanism — alternating between local optimizations within each block of factors and message passing across different blocks. A fundamental tradeoff is between the cost of solving the local optimizations and the induced global convergence rate. A representative work with simple local optimizations is the Projected Subgradient Dual Decomposition (PSDD) algorithm [5]. Each of its local optimization depends on a single factor and optimizes a linear objective under a simplex constraint, hence admits a close-form solution via a simple enumeration. Yet the number of iterations for PSDD to find an \(\epsilon\)-precise solution is \(O(1/\epsilon^2)\). A representative work with faster convergence is the Alternating Direction Dual Decomposition (AD3) method [14] which has \(O(1/\epsilon)\) outer iteration complexity. However, its local optimizations are based on quadratic programings, which usually do not admit closed-form solutions and are solved via costly optimization sub-procedures. The AD3 algorithm exploits an active-set method to reduce the cost spent on the inner iterations, but the cost for variable selection is linear to the factor domain size. In contrast, our active-set method performs only a closed-form update w.r.t. the active set, and utilizes data structures to achieve a sublinear cost of variable selection.

The barrier to fast convergence of the Dual Decomposition techniques is their inherent non-smoothness of dual objectives due to LP relaxations. This non-smoothness also hinders global convergence for algorithms that replace the Subgradient Descent with the Block Coordinate Descent, such as Tree-Reweighted Belief Propagation (TRWBP) [15], Tree-Reweighted max-product message passing with Sequential update(TRW-S) [7], and Max-Product Linear Programming (MPLP) [12]. In consequence, even though the Block-Co-ordinate Descent (TRW-S, MPLP) methods exhibit faster convergence rate, they often suffer from the issue of suboptimal fixed-points.

To achieve the fast convergence rate, a common strategy is to add strongly convex terms to the primal and/or dual objectives [9, 11]. However, such a methodology incurs another layer of relaxation due to these additional terms. Although these approaches still achieve \(O(1/\epsilon)\) iteration complexity (e.g., when the Nerstrov’s acceleration technique is used [16]), their performance heavily relies on adjusting the weights of these additional terms carefully. Moreover, these approaches typically use full-gradient updates which obstruct exploiting the sparsity in the messages (dual variables). One exception is the work of [11] which utilizes the sparse optimizations on the unsmoothed objective function. However, this method only achieves an \(O(1/\epsilon^2)\) iteration complexity.

Note that the per-iteration cost of all existing LP relaxation methods grows linearly with the size of an output domain. Apparently, this becomes intractable when the size of state space increases. In this work, we show how to exploit the sparsity of the messages in large-output-domain problems to achieve a sublinear cost per iteration and guarantee a fast convergence rate. Note a similar strategy could be also exploited at the training phase for some specific loss function such as structured support vector machine [17, 18].

3 Algorithm

In this section, we describe the proposed GDMM algorithm for MAP inference. We begin by describing a variant of the augmented Lagrangian to a linear programming relaxation for the MAP inference problem (Sec. 3.1). We then show how to optimize it efficiently using a greedy direction method of multiplier (Sec. 3.2). We also propose another variant to efficiently solve the problems with binary variables and a large number of factors (Sec. 3.3). For simplicity, we will focus our description on pairwise-interacted factors, but the technique can be easily generalized to higher order factors.

3.1 Augmented Lagrangian for MAP Inference

We study the standard MAP inference problem described by a factor graph \(G = \{\mathcal{V}, \mathcal{F}, \mathcal{E}\}\), where \(\mathcal{V}, \mathcal{F}, \mathcal{E}\) are the sets of random variables, factors, and connecting edges, respectively. We denote the neighboring variables for each factor \(f \in \mathcal{F}\) as \(\partial(f) = \{i | (f, i) \in \mathcal{E}\}\) and similarly the neighboring factors for each variable.
Applying the Augmented Lagrangian method, the goal of the Maximum-a-Posteriori (MAP) problem is to find an assignment \( \{x_i \in X_i | i \in V \} \) (i.e. each variable takes one value) that maximizes the sum of local score functions:

\[
\max_x \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(y_f) \tag{1}
\]

Here \( y_f \in Y_f := \prod_{i \in \partial(f)} X_i \) collects assignments from variables in \( \partial(f) \). Note that the factors associated with only one variable can be absorbed by its neighboring variable, so we assume all factors are of order at least two.

We consider a standard LP relaxation for (1). Specifically, we replace \( x_i \) and \( y_f \) with marginal vectors \( x_i \in \Delta_i = \Delta_i^{|X_i|} \) and \( y_f \in \Delta_f = \Delta_f^{|Y_f|} \), where \( \Delta_i = \{ \alpha_i \in \mathbb{R}^{|X_i|} | a \geq 0, \sum_{i=1}^{|X_i|} a_i = 1 \} \) is a simplex of dimension \( M \). The marginal vectors are highly constrained with each other. The LP relaxations typically relax these constraints as linear constraints. In this paper, we consider standard linear constraints given by \( \forall (f,i) \in E, M_{if} y_f = x_i \), where

\[
M_{if}(x_i, y_f) = \begin{cases} 
1 & \text{if } y_f \sim x_i \\
0 & \text{otherwise} 
\end{cases}
\]

and \( y_f \sim x_i \) means that \( x_i \) is consistent with \( y_f \)’s configuration.

Let \( \theta_i \) and \( \theta_f \) be vector representations of the local score functions, we derive the LP relaxation in this paper:

\[
\max_{x_i \in \Delta_i, y_f \in \Delta_f} \sum_{i \in V} \theta_i^T x_i + \sum_{f \in F} \theta_f^T y_f 
\quad \text{s.t.} \quad \forall (f,i) \in E, M_{if} y_f = x_i \tag{2}
\]

Applying the Augmented Lagrangian method, the objective function becomes (constraints remain the same):

\[
\mathcal{L}(\{x_i\}, \{y_f\}) = \sum_{i \in V} (-\theta_i)^T x_i + \sum_{f \in F} (-\theta_f)^T y_f + \sum_{(f,i) \in E} \left( \frac{\rho}{2} \| M_{if} y_f - x_i + \frac{1}{\rho} \mu_{if} \|_2^2 \right). \tag{3}
\]

For optimization, we alternate between optimizing the primal variables \( \{x_i, y_f\} \) and the dual variables \( \mu_{if} \). Note that the primal constraints \( x_i \in \Delta_i \) and \( y_f \in \Delta_f \) are strictly enforced during optimization.

### Algorithm 1 GDMM for Large Factor Domain

**Initialization:** \( \forall i \in V, x_i \leftarrow 0, A_i \leftarrow \emptyset; \forall f \in F, y_f \leftarrow 0, A_f \leftarrow \emptyset, \forall (f,i) \in E, \mu_{if}^0 \leftarrow 0 \).

repeat

for \( s = 1, 2, \ldots, S \) do

for \( \gamma = 1, 2, 4, 8, \ldots, \gamma_{\text{max}} \) do

1. \( \forall f \in F, \) update \( A_f, y_f(A_f) \) according to (8) and (6) with \( Q = \gamma Q_{\text{max}} \).

2. \( \forall i \in V, \) update \( A_i, x_i(A_i) \) according to (7) and (4).

3. Break if \( \mathcal{L}(x', y') - \mathcal{L}(x, y) \leq \sigma (\nabla_y \mathcal{L}, \Delta y) \)

end for

end for

for \( \forall (f,i) \in E \) do

3. Compute \( \mu_{if}^{s+1} \) based on (9)

end for

4. \( t \leftarrow t + 1 \)

until primal/dual infeasibility \(< \epsilon \)

#### 3.2 Greedy Direction Method of Multiplier for Factor of Large Domain

We proceed to describe the proposed greedy direction method of multiplier (or GDMM) for optimizing (3) (Please refer to Algorithm 1 for details). GDMM alternates between updating the dual variables and sequentially optimizing the individual primal variables \( x_i \) and \( y_f \) w.r.t. a judiciously selected active set of states within the factor. However, the cost of such optimization is still expensive when the label space is large. This leads to the key component of this paper, i.e., by maintaining active sets of coordinates for each variable, one can pass only active messages to the nearby factors. Once the messages become sparse, the selection of active variables can be implemented efficiently using pre-built data structures. These active sets are initialized as empty sets and are gradually augmented in an on-demand manner. Our approach is motivated by the observations that the non-zero entries in \( x_i \) and \( y_f \) are relatively sparse and the size of these active sets is small throughout the optimization. These observations could result in significant speedup of the algorithm. Next, we present each step of the GDMM method in detail: optimizing the primal variables with respect to the active sets, updating the active sets, and updating the dual variables.

**Primal variable optimization.** The primal variables are optimized by the Fully-corrective Frank-Wolfe (FCFW) with approximate correction [19] which alternates between optimizing \( \{x_i\} \) and \( \{y_f\} \) until a certain precision is achieved. For each \( i \in V \), we use \( A_i \) and \( \bar{A}_i \) to denote its active set and the complementary set, respectively. The subproblem of (3) w.r.t. \( A_i \)
is given by
\[
\min_{x_i \in \Delta_i, x_i(\mathcal{A}_i)=0} -\theta_i^T x_i + \frac{\rho}{2} \sum_{f \in \partial(i)} \| M_{if} y_f - x_i + \frac{1}{\rho} \mu_{if}^T\|^2
\]
which is equivalent to the simplex projection problem
\[
\min_{x_i \in \Delta_i, x_i(\mathcal{A}_i)=0} \left\| x_i - \left( \sum_{f \in \partial(i)} \left( \frac{M_{if} y_f}{\rho \partial(i)} + \frac{1}{\rho} \mu_{if}^T + \frac{\theta_i}{\rho \partial(i)} \right) \right) \right\|^2_2
\]
(4)

As in [20], (4) can be solved via a simplex projection operation with time complexity \(O(|\mathcal{A}_i| \log(|\mathcal{A}_i|))\). This also implies that, when fixing \(\mu, x\) can be written as a simple function \(x(y)\).

Now we consider optimizing the variables associated with each factor \(f \in \mathcal{F}\). Let \(\mathcal{A}_f\) and \(\mathcal{A}_f^c\) be its active set and the complementary set. The subproblem w.r.t. \(\mathcal{A}_f\) is
\[
\min_{y_f \in \Delta_f, y_f(\mathcal{A}_f)=0} \quad \mathcal{L}_f(x, y_f; \mu) := -\theta_f^T y_f + \frac{\rho}{2} \sum_{i \in \partial(f)} \| M_{if} y_f - x_i + \frac{1}{\rho} \mu_{if}^T\|^2
\]
(5)

Instead of solving (5) exactly, we propose to minimize a quadratic upper bound of the function (5). Let \(Q = \rho \| M \|^2\) where \(\| \cdot \|\) is the spectral norm. (5) becomes:
\[
\min_{y_f^+ \in \Delta_f, y_f^+(\mathcal{A}_f)=0} \quad \frac{Q}{2} \| y_f^+ - y_f \|^2 + \nabla y_f^T \mathcal{L}^T(\nabla y_f^+ - y_f)
\]
(6)

where \(\nabla y_f \mathcal{L} = \sum_{i \in \partial(f)} M_{if}^T (\mu_{if}^T - \rho x_i) - \theta_f\). Note the quadratic upper bound (6) is tighter than that used in the Proximal Gradient Descent scheme described in [21] by a factor of \(|\mathcal{A}_f| / |\mathcal{X}|\). It can be solved by a simplex projection
\[
\min_{y_f^+ \in \Delta_f, y_f^+(\mathcal{A}_f)=0} \quad \| y_f^+ - y_f \|^2 - \frac{Q}{2} \sum_{i \in \partial(f)} M_{if}^T (\mu_{if}^T - \rho x_i) - \theta_f)
\]

in time \(O(|\mathcal{A}_f| \log(|\mathcal{A}_f|))\). In practice, computing \(Q\) could be expensive, so we introduce a line-search procedure over \(Q\), starting from a lower bound \(Q_{\text{max}}\) where \(Q_{\text{max}} := \max_{f \in \mathcal{F}} Q_{\mathcal{A}_f}\) and \(Q_{\mathcal{A}_f} = \rho \| |M_{if} |_{\mathcal{A}_f}\|^2\). In practice, we found the descent amount often passes the line-search condition without backtracking. Although we introduce a fully-corrective loop \(s = 1 \ldots S\) in Algorithm 1 for ease of our analysis, we found setting \(S = 1\) suffices for fast convergence in our experiments. Note by maintaining a small set of non-zero variables, we limit the size of messages (non-zero dual variables), yielding an efficient scheme for finding new active variables as introduced next.

**Updating the active sets.** The active sets are updated by finding a currently non-active coordinate with largest gradient magnitude. For each \(i \in \mathcal{V}\), we update the corresponding active set as
\[
\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \arg \max_{y_f \not\in \mathcal{A}_i} \nabla x_i \mathcal{L}_i(x, y; \mu)
\]
(7)

where \(\nabla \mathcal{L}_i(x, y; \mu) = -\sum_{f \in \partial(i)} [\rho(M_{if} y_f - x_i) + \mu_{if}^T y_f - \theta_i(x_i)]\). Similarly, for each \(f \in \mathcal{F}\), we update the corresponding active set as
\[
\mathcal{A}_f \leftarrow \mathcal{A}_f \cup \arg \max_{y_f \not\in \mathcal{A}_f} \nabla y_f \mathcal{L}_f(x, y; \mu)
\]
(8)

where \(\nabla \mathcal{L}_f(x, y; \mu) = \sum_{i \in \partial(f)} [M_{if}^T (\rho(M_{if} y_f - x_i) + \mu_{if}^T)] y_{i'} - \theta_f(y_{i'})\). Note that the new active coordinates can be searched efficiently as long as the messages passed from nearby factors \(M_{if} y_f - x_i\) are sparse. Specifically, as \(\theta_i\) and \(\theta_f\) are constant, we can build sorted lists on \(\theta_i, \theta_f\) in the preprocessing phase. When executing our algorithm, the selection (7) can be done in time \(O(|\mathcal{A}_i| + \sum_{f \in \partial(i)} |\mathcal{A}_f|)\) since the size of non-zero messages is bounded by \(\sum_{f \in \partial(i)} |\mathcal{A}_f|\).

The time complexity of the coordinate selection of factor based on (8) is \(O(|\mathcal{A}_f| |\mathcal{A}_f| + |\mathcal{A}_f|)\), where the messages are denoted by \(\delta_{ij} = \rho(M_{if} y_f - x_i) + \mu_{if}^T\). For a pairwise-interaction factor \(f = (i, i') \in \mathcal{F}\) with \(y_{i'} = (x_{i'}, x_{i'})\) and \(i, i' \in \mathcal{V}\), we can find a coordinate of largest gradient magnitude (8) by searching in four divisions of the coordinates: (i) \(\{(x_{i'}, x_{i'}), \delta_{ij}(x_{i'}) = \delta_{ij}(x_{i'}) = 0\}\), (ii) \(\{(x_{i}, x_{i'})|\delta_{ij}(x_{i}) = 0\}\), (iii) \(\{(x_{i}, x_{i'})|\delta_{ij}(x_{i'}) = 0\}\) and (iv) \(\{(x_{i'}, x_{i'})|\delta_{ij}(x_{i'}) \neq 0, \delta_{ij}(x_{i'}) = 0\}\). In particular, (i) can be done in \(O(1)\) by a sort list on \(\theta_f\). (ii), (iii) can be done in time \(O(|\mathcal{A}_f| |\mathcal{A}_f| + |\mathcal{A}_f|)\) via sorted lists built on the sets \(\theta_f(x_i, x_{i'}|x_i = k\})\, \theta_f(x_i, x_{i'}|x_{i'} = k\})\) respectively.

The complexity of (iv) is \(O(|\mathcal{A}_f| |\mathcal{A}_f|)\).

To maintain a compact active set, after solving (4) and (6), we remove all coordinates \(x_i, y_f\) with \(x_i(x_i) = 0, y_f(y_f) = 0\) from \(\mathcal{A}_i, \mathcal{A}_f\), respectively.

**Updating the dual** After optimizing the primal variables, the dual variables are updated as
\[
\mu_{if}^{t+1} = \mu_{if}^t + \eta(M_{if} y_f - x_i), \quad \forall i \in \mathcal{V}, f \in \mathcal{F},
\]
(9)
where \(\eta\) is the dual step size. In the analysis section we show for a sufficiently small \(\eta\), Algorithm 1 globally converges to the optimum.
Algorithm 2 GDMM for Large Number of Factors

Initialization: \( \forall i \in V, x_i \leftarrow 0; \forall f \in F, y_f \leftarrow 0; \forall (f, i) \in E, \mu_{ij}^0 \leftarrow 0; t \leftarrow 0; A_F \leftarrow \emptyset; A_V \leftarrow \emptyset \).

repeat
1. Find \( i^* \) satisfying (10), \( A_V \leftarrow A_V \cup \{ i^* \} \).
2. \( A_F \leftarrow A_F \cup \{ f \in F | \partial(f) \subseteq A_V \} \).
3. \( \forall i \in A_V, \) update \( x_i(X_i) \) according to (4).\( \) 
4. Find \( f^* \) satisfying (11), \( A_F \leftarrow A_F \cup \{ f^* \} \).
5. \( \forall f \in A_F, \) update \( y_f(Y_f) \) according to (6).
6. Update \( \mu_{ij}^{t+1} \) base on (9).
7. \( t \leftarrow t+1 \)
until primal/dual infeasibility < \( \epsilon \)

3.3 GDMM for a Large Number of Factors

For applications such as alignment, segmentation and multilabel prediction, the factor graphs comprise a large number of factors with binary random variables. In this section, we introduce a variant of the GMDD specifically for such problems. The procedure is sketched in Algorithm 2.

The idea behind Algorithm 2 is to maintain two active sets for variables and factors, denoted as \( A_V \subseteq V \) and \( A_F \subseteq F \), respectively. Then we ensure that at each iteration a variable \( i^* \) satisfying

\[
i^* = \arg\min_{i} \min_{x, d \in \Delta_i} \langle \nabla_x \mathcal{L}_i(x, y; \mu), d \rangle + \frac{1}{2} \|d\|^2
\]

(10)

and a factor \( f^* \) satisfying

\[
f^* = \arg\min_{f} \min_{y, d \in \Delta_f} \langle \nabla_y \mathcal{L}_f(x, y; \mu), d \rangle + \frac{Q_f}{2} \|d\|^2
\]

(11)

are in the active sets (i.e. \( i^* \in A_V \) and \( f^* \in A_F \)).

Note that all variables are binary and all factors have only 4 states, i.e. \( \mathcal{Y}_j = \{(0, 0), (0, 1), (1, 0), (1, 1)\} \). Therefore one can always write \( x_i(0) = 1 - x_i(1) \) and write \( y_f(0, 0), y_f(0, 1), y_f(1, 0) \) as a function of \( y_f(1, 1), x_j(1) \) and \( y_f(1, 1) \) by enforcing consistency constraints \( y_f(1, 0) + y_f(1, 1) = x_i(1) \) and \( y_f(0, 1) + y_f(1, 1) = x_j(1) \) for a factor connecting to two variables \( i \) and \( j \). Then one can verify the remaining constraints between factors and variables are:

\[
x_i(1) - y_f(1, 1) = y_f(1, 0) \geq 0
\]
\[
x_j(1) - y_f(1, 1) = y_f(0, 1) \geq 0
\]
\[
1 - x_j(1) - x_i(1) + y_f(1, 1) = y_f(0, 0) \geq 0
\]

(12)

One can encode (12) as the new set of constraints \( \mathcal{M}_f y_f - x_i = 0 \) by introducing slack variables for the inequalities. Under this scheme, we show an efficient method to include all variables and factors that satisfy (10) and (11) in the active sets.

We achieve this by first pushing any inactive \( f \) into \( A_F \) if its neighboring variables \( i \) and \( j \) are both active. Note this accounts for only a small fraction of the factors if \( |A_V| \ll |V| \), and thus can be computed efficiently. Then we consider any inactive factor \( f \) with either \( i \) or \( j \) inactive. For this type of factor, we have (12) satisfied and \( \mu_{ij} = \mu_{ji} = 0 \), and the gradient of each factor \( y_f := y_f(1, 1) \) is simply

\[
\nabla_{y_f} \mathcal{L}_f(x, y; \mu) = -\theta_f(1, 1).
\]

Therefore, the search of active factor (11) reduces to solving the minimal \( \theta_f(1, 1) \) among \( f \not\in A_F \), which can be easily done via a sorted list of \( -\theta_f(1, 1) \). For each iteration, we add the first inactive \( f \) from the head until we have \( -\theta_f(1, 1) > 0 \) which can be done with \( O(1) \) amortized cost. In our experiments, only a small fraction of \( F \) will be added to \( A_F \). One can refer to Appendix 9 for the statistics of active size \( |A_F| \) in our experiments.

4 Convergence Analysis

In this section, we show the GDMM algorithms for the large factor domains (Algorithm 1) and a large number of factors (Algorithm 2) admit linear convergence. To state the main result formally, we introduce the following notations. We use \( z^t = (x^t, y^t) \) to encapsulate the primal variables, and denote the feasible space as \( \mathcal{M} := \{(x, y) \in \Gamma | x_i \in \Delta_i, \forall i \in V; y_f \in \Delta_f, \forall f \in F \} \). Then the Augmented Lagrangian is expressed as

\[
\mathcal{L}(z, \mu) = (-\theta + M^T \mu, z) + \frac{\rho}{2} \|Mz\|^2,
\]

where \( Mz = 0 \) collects all the constraints of the form

\[
M_{ij} y_f - x_i = 0, \forall (i, f) \in \mathcal{E}.
\]

To assess the convergence rate, denote a primal minimizer respect to a dual iterate \( \mu^t \) at iteration \( t \) as

\[
\tilde{z}^t := \arg\min_{z \in \mathcal{M}} \mathcal{L}(z, \mu^t).
\]

(13)

The convergence rate is measured by the primal gap \( \Delta^t_p \) and the dual gap \( \Delta^t_d \):

\[
\Delta^t_p := \mathcal{L}(z^{t+1}, \mu^t) - \mathcal{L}(\tilde{z}^t, \mu^t)
\]
\[
\Delta^t_d := d^* - \mathcal{L}(\tilde{z}^t, \mu^t)
\]

where \( d^* := \max_{\mu \in \mathcal{M}} \min_{z \in \mathcal{M}} \mathcal{L}(z, \mu) \) is the optimal dual objective value. The following part states two major results for the convergence of the GDMM in the form of both Algorithm 1 and Algorithm 2.
Theorem 1 (Convergence of Algorithm 1). Let $Q = \rho \| M \|^2$. For any constant dual step size $\eta$ satisfying

$$0 < \eta \leq \frac{\rho}{4(1 + |F|/m_M)},$$

the iterates given by Algorithm 1 obey

$$\Delta^t_p + \Delta^t_q \leq \epsilon, \forall t \geq \max \left\{ 2(1 + \frac{|F|}{m_M}) \frac{\tau}{\eta} \right\} \log \left( \frac{1}{\epsilon} \right).$$

Here $m_M$ is the generalized geometric strong convexity constant of function $L(\cdot, \mu)$ on the domain $M$ (defined in Lemma 5), and $\tau$ is a constant characterized in Lemma 4.

Theorem 2 (Convergence of Algorithm 2). Let $Q_{\text{max}} := \max_{f \in \mathcal{F}} Q_f$. For any constant dual step size $\eta$ satisfying

$$0 < \eta \leq \frac{1}{4(1 + Q_{\text{max}}/m_1)},$$

the iterates given by Algorithm 2 have

$$\Delta^t_p + \Delta^t_q \leq \epsilon, \forall t \geq \max \left\{ 2(1 + \frac{Q_{\text{max}}}{m_1}) \frac{\tau}{\eta} \right\} \log \left( \frac{1}{\epsilon} \right),$$

where $m_1$ is a generalized strong convexity constant of function $L(\cdot, \mu)$ measured in $\ell_1$-norm (defined in Lemma 4).

Theorem 1 and 2 are based on the analysis framework of [22] for the Augmented Lagrangian Method, as well as the recent results on the linear convergence of Frank-Wolfe variants [23] and Greedy Coordinate Descent for non-strongly convex functions of the form (13) [24, 25]. A proof outline can be found in Appendix A followed by a detailed proof in Appendix B.

## 5 Experimental Evaluation

Here we evaluate the GDMM algorithm experimentally. Section 5.1 discusses the experimental setup. Section 5.2 analyzes the performance of the GDMM and compare it with state-of-the-art MAP inference techniques.

### 5.1 Experimental Setup

**Benchmark datasets.** We select five datasets that involve large output domains. These datasets are selected from two benchmarks: OPENGM\(^\dagger\) and The Probabilistic Inference Challenge 2011 (PASCAL)\(^\ddagger\). As shown in Table 1, they cover a diverse set of examples with varying scales, network connectivities, and output domain sizes. To help understand the performance of the algorithms, we briefly describe each dataset:

- **Protein.** [26, 27] A Protein Folding dataset that is included in both PASCAL and OpenGM. Its factor graph has bigram factors between every pair of variables.
- **Graph Matching.** A real world social network matching dataset generated using Facebook network data from SNAP\(^\ddagger\). We attempt to match subgraphs to the original graph. In particular, we construct the factor graph as follows: 1. one variable for each node in a subgraph, with vertex set of the original graph as its domain. We use inner product of the features provided from SNAP to generate the unigram factors. 2. For each induced edge on the subgraph, we introduce a bigram factor for the two variables and use inverse of the shortest distance on the graph to generate the bigram factor.
- **Segmentation.** A dataset from PASCAL which has a grid-4 neighborhood structure.
- **ImageAlignment.** A dataset from PASCAL.
- **Multilabel.** A multilabel data set from Mulan\(^\ddagger\). The MAP inference problem is generated from a trained model and an instance from the testset.

**Compared Algorithms.** The compared algorithms include several top performing algorithms as well as widely used MAP inference algorithms.

| Dataset        | $|X_i|$ | $|Y_i|$ | $|V|$ | $|F|$ |
|----------------|--------|--------|------|------|
| Segmentation   | 21     | 441    | 226  | 842  |
| ImageAlignment | 83     | 6889   | 400  | 3334 |
| MultiLabel     | 2      | 4      | 3884 | 7544670 |
| Protein        | 404    | 163216 | 37   | 703  |
| GraphMatching  | 1034   | 1069156| 188  | 1864 |

Table 1: Data Statistics \(^\dagger\). $|X_i|$ denotes the maximum domain size of an output variable, $|Y_i|$ is the maximum domain size of a factor, and $|V|, |F|$ denote the number of nodes, factors respectively.

---

\(^\dagger\)The specific instances we used for each dataset are: 16,16,8,21.uai (Segmentation), fileforGal400markers.uai (ImageAlignment), EurLex (Multilabel), and 2BBN.uai (Protein)

\(^\ddagger\)http://hciweb2.iwr.uni-heidelberg.de/opengm/

\(^\ddagger\)http://www.cs.huji.ac.il/project/PASCAL/showNet.php

\(^\ddagger\)http://snap.stanford.edu

\(^\ddagger\)http://mulan.sourceforge.net/datasets-mlc.html
Table 2: Primal: best decoded (integer solution) primal objective; Time: time taken to reach the decoded primal objective. The shortest time taken to reach the best MAP objective among all methods are marked with boldface. For all experiments we set memory limit = 100G and time limit = 2 day. Experiments violating these limits are marked MLE: Memory Limit Exceeded, or TLE: Time Limit Exceeded; NE means the solver throws error message “quantity not normalizable”.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Algorithm</th>
<th>Segmentation Time</th>
<th>Primal</th>
<th>Image/Alignment Time</th>
<th>Primal</th>
<th>Multilabel Time</th>
<th>Primal</th>
<th>Protein Time</th>
<th>Primal</th>
<th>Graph Matching Time</th>
<th>Primal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LBP</td>
<td>0.406s</td>
<td>-252.39</td>
<td>4.136s</td>
<td>-6907</td>
<td>5012.6s</td>
<td>899.06</td>
<td>86407s</td>
<td>11904</td>
<td>17606s</td>
<td>3733</td>
</tr>
<tr>
<td></td>
<td>TRWBP</td>
<td>1.6336s</td>
<td>-252.39</td>
<td>17.2s</td>
<td>-6907</td>
<td>25474s</td>
<td>899.06</td>
<td>75205s</td>
<td>11888</td>
<td>1598s</td>
<td>3733</td>
</tr>
<tr>
<td></td>
<td>TreeEP</td>
<td>11.514s</td>
<td>-252.39</td>
<td>2666s</td>
<td>-6016.95</td>
<td>2863.2s</td>
<td>899.06</td>
<td>2010.4s</td>
<td>11974</td>
<td>TLE</td>
<td>3733</td>
</tr>
<tr>
<td></td>
<td>HAK</td>
<td>6.75s</td>
<td>-252.39</td>
<td>274.5s</td>
<td>-6908</td>
<td>TLE</td>
<td>TLE</td>
<td>10013s</td>
<td>10740</td>
<td>87886s</td>
<td>3717</td>
</tr>
<tr>
<td></td>
<td>GBP</td>
<td>0.22s</td>
<td>-252.39</td>
<td>6.07s</td>
<td>-6908</td>
<td>TLE</td>
<td>TLE</td>
<td>NE</td>
<td>NE</td>
<td>3432s</td>
<td>3721</td>
</tr>
<tr>
<td></td>
<td>SoftBCFW</td>
<td>20.89s</td>
<td>-356.57</td>
<td>597.7s</td>
<td>-19572</td>
<td>54.22s</td>
<td>889.32</td>
<td>1838s</td>
<td>9250</td>
<td>MLE</td>
<td>MLE</td>
</tr>
<tr>
<td></td>
<td>LPsparse</td>
<td>0.57s</td>
<td>-252.39</td>
<td>370s</td>
<td>-6913.1</td>
<td>N/A</td>
<td>N/A</td>
<td>32000s</td>
<td>12179</td>
<td>MLE</td>
<td>MLE</td>
</tr>
<tr>
<td></td>
<td>SmoothMSD</td>
<td>11.19s</td>
<td>-252.39</td>
<td>6462s</td>
<td>-6907.6</td>
<td>53462s</td>
<td>893.66</td>
<td>13648s</td>
<td>53462s</td>
<td>899.06</td>
<td>3733</td>
</tr>
<tr>
<td></td>
<td>MPLP</td>
<td>0.12s</td>
<td>-252.39</td>
<td>9.85s</td>
<td>-6907.6</td>
<td>4514s</td>
<td>846.64</td>
<td>124.3s</td>
<td>11771</td>
<td>212.5s</td>
<td>3730</td>
</tr>
<tr>
<td></td>
<td>PSDD</td>
<td>2.596s</td>
<td>-319.38</td>
<td>531.4s</td>
<td>-6907.6</td>
<td>1503s</td>
<td>899.06</td>
<td>20865s</td>
<td>10013s</td>
<td>MLE</td>
<td>MLE</td>
</tr>
<tr>
<td></td>
<td>AD$^3$</td>
<td>0.15s</td>
<td>-252.39</td>
<td>60.2s</td>
<td>-6917.09</td>
<td>243.03s</td>
<td>899.06</td>
<td>48137s</td>
<td>12000.8</td>
<td>MLE</td>
<td>MLE</td>
</tr>
<tr>
<td></td>
<td>TRW-S</td>
<td>0.0046s</td>
<td>-252.39</td>
<td>0.65s</td>
<td>-6907</td>
<td>21.89s</td>
<td>899.06</td>
<td>39.67s</td>
<td>12067</td>
<td>47.3s</td>
<td>3732</td>
</tr>
<tr>
<td></td>
<td>GDMN</td>
<td>0.26s</td>
<td>-252.39</td>
<td>18.09s</td>
<td>-6907.6</td>
<td>16.02s</td>
<td>899.06</td>
<td>661s</td>
<td>12263s</td>
<td>25.9s</td>
<td>3733</td>
</tr>
</tbody>
</table>

Overall, GDMM is the top-performing algorithm. It returns the best solutions among all five datasets. In terms of running time, it is also competent with the top algorithms on each dataset. On the grid-4 structured factor graphs such as Segmentation, TRWS and MPLP are generally fast and accurate. But they are also less accurate on several exam-
In this paper, we introduce GDMM, an efficient algorithm for MAP inference of large output domain. GDMM combines the power of greedy method (FW, GCD) with ALM to achieve a cost sublinear to the domain size. We show that despite the usage of active sets, the algorithm still admits a linear-type convergence rate, and experimental results on benchmark datasets reveal superiority of the algorithm on large-scale problems. As a future work, we plan to develop algorithm that integrates our approach with the fast convergence of TRWS.

Acknowledgement Q.H. acknowledges the support of NSF via DMS-1700234. I.D. acknowledges the support of NSF via CCF-1320746, IIS-1546452, and CCF-1564000. P.R. acknowledges the support of ARO via W911NF-12-1-0390 and NSF via IIS-1149803, IIS-1320894, IIS-1447574, DMS-1264033, and NIH via R01 GM117594-01 as part of the Joint DMS/NIGMS Initiative to Support Research at the Interface of the Biological and Mathematical Sciences.
References


