2-Slave Dual Decomposition for Generalized Higher Order CRFs

Xian Qian and **Yang Liu**
Computer Science Department
The University of Texas at Dallas
{qx,yangl}@hlt.utdallas.edu

Abstract

We show that the decoding problem in generalized Higher Order Conditional Random Fields (CRFs) can be decomposed into two parts: one is a tree labeling problem that can be solved in linear time using dynamic programming; the other is a supermodular quadratic pseudo-Boolean maximization problem, which can be solved in cubic time using a minimum cut algorithm. We use dual decomposition to force their agreement. Experimental results on Twitter named entity recognition and sentence dependency tagging tasks show that our method outperforms spanning tree based dual decomposition.

1 Introduction

Conditional Random Fields (Lafferty et al., 2001) (CRFs) are popular models for many NLP tasks. In particular, the linear chain CRFs explore local structure information for sequence labeling tasks, such as part-of-speech (POS) tagging, named entity recognition (NER), and shallow parsing. Recent studies have shown that the predictive power of CRFs can be strengthened by breaking the locality assumption. They either add long distance dependencies and patterns to linear chains for improved sequence labeling (Galley, 2006; Finkel et al., 2005; Kazama and Torisawa, 2007), or directly use the 4-connected neighborhood lattice (Ding et al., 2008). The resulting non-local models generally suffer from exponential time complexity of inference except some special cases (Sarawagi and Cohen, 2004; Takhanov and Kolmogorov, 2013; Kolmogorov and Zabih, 2004).

Approximate decoding algorithms have been proposed in the past decade, such as reranking (Collins, 2002b), loopy belief propagation (Sutton and Mccallum, 2006), tree reweighted belief propagation (Kolmogorov, 2006). In this paper, we focus on dual decomposition (DD), which has attracted much attention recently due to its simplicity and effectiveness (Rush and Collins, 2012). In short, it decomposes the decoding problem into several sub-problems. For each sub-problem, an efficient decoding algorithm is deployed as a slave solver. Finally a simple method forces agreement among different slaves. A popular choice is the sub-gradient algorithm. Martins et al. (2011b) showed that the success of the sub-gradient algorithm is strongly tied to the ability of finding a good decomposition, i.e., one involving few overlapping slaves. However, for generalized higher order graphical models, a lightweight decomposition is not at hand and many overlapping slaves may be involved. Martins et al. (2011b) showed that the sub-gradient algorithm exhibits extremely slow convergence in such cases, and they proposed the alternating directions method (DD-ADMM) to tackle these.

In this paper, we propose a 2-slave dual decomposition approach for efficient decoding in higher order CRFs. One slave is a tree labeling model that can be solved in linear time using dynamic programming. The other is a supermodular quadratic pseudo-Boolean maximization problem, which can be solved in cubic time via minimum

Transactions of the Association for Computational Linguistics, 2 (2014) 339–350. Action Editor: Kristina Toutanova.
Submitted 11/2013; Revised 6/2014; Published 10/2014.

cut. Experimental results on Twitter NER and sentence dependency tagging tasks demonstrate the effectiveness of our technique.

2 Background

2.1 Generalized Higher Order CRFs

Given an undirected graph $G = (V, E)$ with N vertices, let $\mathbf{x} = x_1, x_2, \ldots, x_N$ denote the observations of the vertices, and each observation x_v is asked to assign one state (or label) in the state set $s \in \mathcal{S}$. The assignment of the graph can be represented by a binary matrix $Y_{N\times|\mathcal{S}|}$, where $|\mathcal{S}|$ is the cardinality of $\mathcal{S}$, and the element $Y_{v,s}$ indicates if x_v is assigned state s. In the rest of the paper, we use $Y_{v[s]}$ instead, and $v[s]$ to denote the vertex v with state s. The constraint

$$\sum_{s} Y_{v[s]} = 1 \tag{1}$$

is required so that each vertex has exactly one state. In this paper, we use $\mathcal{Y}$ to denote the space of state assignments.

The decoding problem is to search the optimal assignment that maximizes the scoring function

$$Y^* = \arg\max_{Y\in\mathcal{Y}(\mathbf{x})} \phi(\mathbf{x}, Y)$$

where $\phi(\mathbf{x}, Y)$ is a given scoring function. As $\mathbf{x}$ is constant in this maximization problem, we omit $\mathbf{x}$ for simplicity in the remainder of the paper. The decoding problem becomes

$$\max_{Y\in\mathcal{Y}} \phi(Y). \tag{2}$$

The scoring function $\phi(Y)$ is usually decomposed into small parts

$$\phi(Y) = \sum_{\mathbf{c}} \sum_{\mathbf{s}\in\mathbf{c}[\cdot]} \phi_{\mathbf{c}[\mathbf{s}]} \prod_{v[s]\in\mathbf{c}[\mathbf{s}]} Y_{v[s]}$$

where $\mathbf{c}$ is a subset of vertices, called a **factor**. $\mathbf{c}[\cdot]$ is the set of all possible assignments of $\mathbf{c}$. For example, factor $\mathbf{c} = \{u, v\}$ denotes the edge (u, v) in the graph, and $\mathbf{c}[\cdot] = \mathcal{S}^2$ is the set of the $|\mathcal{S}|^2$ transitions. A factor $\mathbf{c}$ with a specific state assignment $\mathbf{s}$ is called a **pattern**, denoted as $\mathbf{c}[\mathbf{s}]$. For example, $v[s]$ is a pattern of vertex v, and $uv[st]$ is a pattern of edge (u, v) as shown in Figure 1. Note that our definition extends of the work of Takhanov and Kolmogorov where patterns are restricted to the state sequences of consecutive vertices (Takhanov and Kolmogorov, 2013). $\prod_{v[s]\in\mathbf{c}[\mathbf{s}]} Y_{v[s]}$ means a pattern $\mathbf{c}[\mathbf{s}]$ is selected in the assignment only if all its members $v[s]$ are selected. For simplicity, in this paper we use

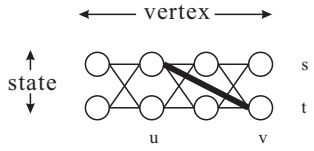


Figure 1: Pattern $\mathbf{c}[\mathbf{s}] = uv[st]$ is shown in bold.

$$Y_{\mathbf{c}[\mathbf{s}]} = \prod_{v[s]\in\mathbf{c}[\mathbf{s}]} Y_{v[s]}$$

to denote whether pattern $\mathbf{c}[\mathbf{s}]$ appears in the assignment. Then the scoring function becomes

$$\phi(Y) = \sum_{\mathbf{c}} \sum_{\mathbf{s}\in\mathbf{c}[\cdot]} \phi_{\mathbf{c}[\mathbf{s}]} Y_{\mathbf{c}[\mathbf{s}]}. \tag{3}$$

Many existing CRFs can be represented using Eq (3). For example, the popular linear chain CRFs consider two types of patterns: vertices and edges connecting adjacent vertices, resulting in the following scoring function

$$\phi(Y) = \sum_{v}\sum_{s} \phi_{v[s]} Y_{v[s]} + \sum_{v} \sum_{st\in\mathcal{S}^2} \phi_{v(v+1)[st]} Y_{v(v+1)[st]}$$

The optimal Y can be found in linear time using the Viterbi algorithm.

Another example is the skip-chain CRFs, which consider the interactions between similar vertices

$$\begin{aligned}\phi(Y) \;=\; & \sum_{v}\sum_{s} \phi_{v[s]} Y_{v[s]} \\ & + \sum_{v} \sum_{st\in\mathcal{S}^2} \phi_{v(v+1)[st]} Y_{v(v+1)[st]} \\ & + \sum_{u,v \text{ are similar}} \sum_{s\in\mathcal{S}} \phi_{uv[ss]} Y_{uv[ss]}.\end{aligned}$$

With positive $\phi_{uv[ss]}$, the model encourages similar vertices u and v to have identical state s, and thus it yields a more consistent labeling result compared with linear chain CRFs. Empirically, the use of complex patterns achieves better performance but suffers from high computational complexity of inference, which is generally NP-hard. Hence an efficient approximate inference algorithm is required to balance the trade-off.

2.2 Dual Decomposition

Dual decomposition is a popular approach due to its simplicity and effectiveness, and has been successfully applied to many tasks such as machine translation, cross sentential POS tagging, joint POS tagging and parsing.

Briefly, dual decomposition attempts to solve problems of the following form

$$\max_{Y} \sum_{i=1}^{M} \phi_i(Y)$$

The objective function is the sum of several small components that are tractable in isolation but whose combination is not. These components are called slaves. Rather than solving the problem directly, dual decomposition considers the equivalent problem

$$\begin{aligned} \max_{Y, Z_1 \ldots Z_M} \quad & \sum_{i=1}^{M} \phi_i(Z_i) \\ \text{s.t.} \quad & Z_i = Y \quad \forall i \end{aligned}$$

Using Lagrangian relaxation to eliminate the constraint, we get

$$\min_{\lambda} \max_{Y, Z_1 \ldots Z_M} \sum_{i=1}^{M} \phi_i(Z_i) + \sum_{i} \lambda_i^T (Y - Z_i) \qquad (4)$$

which provides the upper bound of the original problem. λ is the Lagrange multiplier, which is typically optimized via sub-gradient algorithms.

Martins et al. (2011b) showed that the success of sub-gradient algorithms is strongly tied to the ability of finding a good decomposition, i.e., one involving few slaves. Finding a concise decomposition is usually task dependent. For example, Koo et al. (2010) introduced dual decomposition for parsing with non-projective head automata. They used only two slaves: one is the arc-factored model, and the other is head automata which involves adjacent siblings and can be solved using dynamic programming in linear time.

Dual decomposition is especially efficient for joint learning tasks because a concise decomposition can be derived naturally where each slave solves one subtask. For example, Rush et al. (2010) used two slaves for integrated phrase-structure parsing and trigram POS tagging task.

However, for generalized higher order CRFs, a lightweight decomposition may be not at hand. Martins et al. (2011a) showed that the sub-gradient algorithms exhibited extremely slow convergence when handling many slaves. For fast convergence, they employed alternating directions dual decomposition (AD^3), which relaxes the agreement constraint via augmented Lagrangian Relaxation, where an additional quadratic penalty term was added into the Lagrangian (Eq (4)). Similarly, Jojic et al. (2010) added a strongly concave term to the Lagrangian to make it differentiable, resulting in fast convergence.

The work most closely related to ours is the work by Komodakis (2011), where dual decomposition was used for decoding general higher order CRFs. Komodakis achieved great empirical success even with the naive decomposition where each slave processes a single higher order factor. His result demonstrates the effectiveness of the dual decomposition framework. Our work improves Komodakis' by using a concise decomposition with only two slaves.

2.3 Graph Representable Pseudo-Boolean Optimization

One slave in our approach is a graph representable pseudo-Boolean maximization problem, which can be reduced to a supermodular quadratic pseudo-Boolean maximization problem and solved efficiently using an algorithm for finding a minimal cut.

A pseudo-Boolean function (PBF) (Boros and Hammer, 2002) is a multilinear function of binary variables, that is

$$f(\mathbf{x}) = \sum_{i} a_i x_i + \sum_{i<j} a_{ij} x_i x_j + \sum_{i<j<k} a_{ijk} x_i x_j x_k + \ldots$$

where $x_i \in \{0, 1\}$. Maximizing a PBF is usually NP-hard, such as the maximum cut problem (Boros and Hammer, 1991).

A pseudo-Boolean function is said to be supermodular iff

$$f(\mathbf{x}) + f(\mathbf{y}) \leq f(\mathbf{x} \wedge \mathbf{y}) + f(\mathbf{x} \vee \mathbf{y})$$

where $\mathbf{x} \wedge \mathbf{y}$, $\mathbf{x} \vee \mathbf{y}$ are the element-wise AND and OR operator of the two vectors respectively. This is an important concept, because a supermodular pseudo-Boolean function (SPBF) can be maximized in $O(n^6)$ running time (Orlin, 2009). A necessary and sufficient condition for identifying a SPBF is

that all of its second order derivatives are non-negative (Nemhauser et al., 1978), i.e., for all $i < j$,

$$\frac{\partial f}{\partial x_i \partial x_j} \geq 0$$

For example a quadratic PBF is supermodular if its coefficients of all quadratic terms are non-negative.

Though the general supermodular maximization algorithm can be used for any SPBF, the special features of some specific problems allow more efficient algorithms to be used. For example, it is well known that the supermodular quadratic pseudo-Boolean maximization problem can be solved in cubic time using min-cut (Billionnet and Minoux, 1985; Kolmogorov and Zabih, 2004).

In fact, a subset of SPBFs can be maximized using a min-cut algorithm. A pseudo-Boolean function $f(\mathbf{x})$ is called **graph representable** or graph expressible if there exists a graph $G = (V, E)$ with terminals s and t and a subset of vertices $V_0 = V - \{s, t\} = \{v_1, \ldots, v_n, u_1, \ldots, u_m\}$ such that, for any configuration $x_1, \ldots, x_n$, the value of the function $f(\mathbf{x})$ is equal to a constant plus the cost of the minimum s-t cut among all cuts, in which v_i is connected with s if $x_i = 0$ and connected with t if $x_i = 1$.

Our definition extends the work of Kolmogorov and Zabih (2004) that focused on quadratic and cubic functions. Vertices $u_1, \ldots u_m$ correspond to the extra binary variables that are introduced to reduce the graph representable PBFs to equivalent quadratic forms. For example, the positive-negative PBFs where all terms of degree 2 or more have positive coefficients are graph representable, and each non-linear term requires one extra binary variable to obtain the equivalent quadratic form (Rhys, 1970).

3 The Tree-Cut Decomposition for Generalized Higher Order CRFs

We decompose the decoding problem, i.e., maximization of Eq (3), into two parts, a tree labeling problem and a PBF maximization problem. We show that the PBF can be graph representable by reparameterizing the scoring function in Eq (3). Then we reduce these pseudo-Boolean functions to quadratic forms based on the recent work of Živný and Jeavons (2010), and finally solve the slave problem via graph cuts.

3.1 Fully Connected Pairwise CRFs

We first describe our idea for a simple case, the fully connected pairwise CRFs (Krähenbühl and Koltun, 2011), which are generalizations of linear-chain CRFs and skip-chain CRFs. Formally, the decoding problem in fully connected pairwise CRFs can be formulated as follows:

$$\begin{aligned} \max_{Y} \quad & \sum_{v}\sum_{s} \phi_{v[s]} Y_{v[s]} + \sum_{u,v}\sum_{st\in\mathcal{S}^2} \phi_{uv[st]} Y_{uv[st]} \\ \text{s.t.} \quad & Y \in \mathcal{Y} \end{aligned} \tag{5}$$

Note that for any edge (u, v), adding a constant ψ_{uv} to all of its related patterns will not change the optimal solution of the problem. In other words, the optimal Y for the following problem is irrelevant to ψ_{uv}

$$\begin{aligned} \max_{Y} \quad & \sum_{v}\sum_{s} \phi_{v[s]} Y_{v[s]} \\ & + \sum_{u,v}\sum_{st\in\mathcal{S}^2} (\psi_{uv} + \phi_{uv[st]}) Y_{uv[st]} \\ \text{s.t.} \quad & Y \in \mathcal{Y} \end{aligned}$$

The reparameterization keeps the optimality of the problem and plays an important role for graph representation, as we will show later.

By introducing a new variable $Z = Y$ for the quadratic terms and relaxing the constraint $Z = Y$ using Lagrangian relaxation, we get the relaxed problem

$$\begin{aligned} \min_{\lambda}\max_{Y,Z} \quad & \sum_{v}\sum_{s} \phi_{v[s]} Y_{v[s]} \\ & + \sum_{u,v}\sum_{st\in\mathcal{S}^2} (\psi_{uv} + \phi_{uv[st]}) Z_{uv[st]} \\ & + \sum_{v}\sum_{s} \lambda_{v[s]} \left(Z_{v[s]} - Y_{v[s]}\right) \\ \text{s.t.} \quad & Y \in \mathcal{Y} \\ & Z_{v[s]} \in \{0, 1\}, \quad \forall v, s \end{aligned}$$

We split the inner $\max_{Y,Z}$ into two subproblems, and a minimal λ is found using sub-gradient descent algorithms which repeatedly find a maximizing assignment for the subproblems individually.

Let

$$\begin{aligned} f_\lambda(Y) &= \sum_{v,s} \phi_{v[s]} Y_{v[s]} - \sum_{v,s} \lambda_{v[s]} Y_{v[s]} \\ g_\lambda(Z) &= \sum_{u,v} \sum_{st \in \mathcal{S}^2} (\psi_{uv} + \phi_{uv[st]}) Z_{uv[st]} \\ &\quad + \sum_{v,s} \lambda_{v[s]} Z_{v[s]} \end{aligned}$$

The two subproblems are

$$\begin{aligned} \max_{Y} \quad & f_\lambda(Y) \\ \text{s.t.} \quad & Y \in \mathcal{Y} \end{aligned}$$

and

$$\begin{aligned} \max_{Z} \quad & g_\lambda(Z) \\ & Z_{v[s]} \in \{0,1\}, \quad \forall v, s \end{aligned}$$

The first subproblem can be solved in linear time since all vertices are independent. The second problem is a binary quadratic programming problem. As discussed in Section 2.3, $g_\lambda(Z)$ can be solved using min-cut if the coefficients of the quadratic terms are non-negative, i.e.

$$\psi_{uv} + \phi_{uv[st]} \geq 0, \quad \forall u, v, s, t$$

Hence, we can set

$$\psi_{uv} = - \min_{st \in uv[\cdot]} \{\phi_{uv[st]}\}$$

to guarantee the non-negativity. This supermodular binary quadratic programming problem can be solved via the push-relabel algorithm (Goldberg, 2008) in $O(|\mathcal{S}|N)^3$ running time.

Though Z may not satisfy the constraint $Z \in \mathcal{Y}$ after sub-gradient descent based optimization, Y must satisfy $Y \in \mathcal{Y}$, hence we could use Y as the final solution if Z and Y disagree.

3.2 Generalized Higher Order CRFs

Now we consider the general case, maximizing Eq (3). Similar with the pairwise case, we use two slaves. One is a set of independent vertices, and the other is a pseudo-Boolean optimization problem. That is, we can redefine $g_\lambda(Z)$ as

$$\begin{aligned} g_\lambda(Z) &= \sum_{\mathbf{c}} \sum_{\mathbf{s} \in \mathbf{c}[\cdot]} \left(\psi_{\mathbf{c}} + \phi_{\mathbf{c}[\mathbf{s}]}\right) Z_{\mathbf{c}[\mathbf{s}]} \\ &\quad + \sum_{v,s} \lambda_{v[s]} Z_{v[s]} \end{aligned}$$

A sufficient condition for $g_\lambda(Z)$ to be graph representable is that coefficients of all non-linear terms are non-negative (Freedman and Drineas, 2005). Hence, we can set

$$\psi_{\mathbf{c}} = - \min_{\mathbf{s} \in \mathbf{c}[\cdot]} \{\phi_{\mathbf{c}[\mathbf{s}]}\}$$

to guarantee the non-negativity.

In real applications, higher order patterns are sparse, i.e., $\left|\{\mathbf{s} \in \mathbf{c}[\cdot] \mid \phi_{\mathbf{c}[\mathbf{s}] \neq 0}\}\right| \ll |\mathcal{S}|^{|\mathbf{c}|}$ (Qian et al., 2009; Ye et al., 2009). Hence we could skip the patterns with zero weights ($\phi_{\mathbf{c}[\mathbf{s}]} = 0$) when calculating $\sum_{\mathbf{s} \in \mathbf{c}[\cdot]} \phi_{\mathbf{c}[\mathbf{s}]} Z_{\mathbf{c}[\mathbf{s}]}$ for fast inference. However, the reparameterization described above may introduce many non-zero terms which destroy the sparsity. For example, in the NER task, a binary feature is defined as true if a word subsequence matches a location name in a gazetteer. Suppose $\mathbf{c}$ =*Little York village* is such a word subsequence, then among $|\mathcal{S}|^3$ possible assignments of $\mathbf{c}$, only the one that labels $\mathbf{c}$ =*Little York village* as a location name has non-zero weight. However, the reparameterization may add $\psi_{\mathbf{c}}$ to the other $|\mathcal{S}|^3 - 1$ assignments, yielding many new patterns.

Therefore, we use another reparameterization strategy that exploits the sparsity for efficient decomposition. We only reparameterize the weights of edges, i.e., quadratic terms. Let

$$\begin{aligned} g_\lambda(Z) &= \sum_{\substack{\mathbf{c} \\ |\mathbf{c}|=2}} \sum_{\mathbf{s} \in \mathbf{c}[\cdot]} \left(\psi_{\mathbf{c}} + \phi_{\mathbf{c}[\mathbf{s}]}\right) Z_{\mathbf{c}[\mathbf{s}]} \\ &\quad + \sum_{\substack{\mathbf{c} \\ |\mathbf{c}|\geq 3}} \sum_{\mathbf{s} \in \mathbf{c}[\cdot]} \phi_{\mathbf{c}[\mathbf{s}]} Z_{\mathbf{c}[\mathbf{s}]} + \sum_{v,s} \lambda_{v[s]} Z_{v[s]} \end{aligned}$$

The optimal solution is unchanged for any ψ.

In Appendix A, we show that by setting a sufficiently large ψ, $g_\lambda(Z)$ is graph representable. Such reparameterization method requires at most $N^2|\mathcal{S}|^2$ new patterns $\psi_{\mathbf{c},|\mathbf{c}|=2}$ to make $g_\lambda(Z)$ graph representable. It preserves the sparsity of higher order patterns, hence is more efficient than the naive approach.

3.3 Tree-Cut Decomposition

In some cases, the graph is built by adding sparse global patterns to local models like trees, resulting in nearly tree-structured CRFs. For example, Sutton and Mccallum (2006) used skip-chain CRFs for NER, where skip-edges connecting identical words

were added to linear chain CRFs. Since the skip-edges are sparse, the resulting graphical models are nearly linear chains. To handle the edges in local models efficiently, we reformulate the decomposition. Let $\mathcal{T}$ be a spanning tree of the graph, if edge $(u, v) \in \mathcal{T}$, we put its related patterns into the first slave, otherwise we put its related patterns into the second slave.

For clarity, we formulate the tree-cut decomposition for generalized higher order CRFs. The first slave involves the patterns covered by the spanning tree $\mathcal{T}$, and its scoring function is

$$f_\lambda(Y) = \sum_v \sum_s \phi_{v[s]} Y_{v[s]} - \sum_{v,s} \lambda_{v[s]} Y_{v[s]} + \sum_{\substack{\mathbf{c} \in \mathcal{T} \\ |\mathbf{c}|=2}} \sum_{\mathbf{s} \in \mathbf{c}[\cdot]} \left(\psi_{\mathbf{c}} + \phi_{\mathbf{c}[\mathbf{s}]} + \sum_{\substack{\mathbf{c}'[\mathbf{s}'] \supseteq \mathbf{c}[\mathbf{s}] \\ |\mathbf{c}'|\geq 3, \phi_{\mathbf{c}'[\mathbf{s}']}<0}} \phi_{\mathbf{c}'[\mathbf{s}']} \right) Y_{\mathbf{c}[\mathbf{s}]}.$$

The second slave involves the rest patterns. To get its quadratic form, for each pattern $\mathbf{c}[\mathbf{s}], |\mathbf{c}| = 3$, we introduce one extra binary variable $u_{\mathbf{c}[\mathbf{s}]}$, and for each pattern $\mathbf{c}[\mathbf{s}], |\mathbf{c}| \geq 4$, we introduce $|\mathbf{c}| - 3$ extra binary variables $u^k_{\mathbf{c}[\mathbf{s}]}, k = 0, \ldots, |\mathbf{c}| - 4$. Let $\mathbf{u}$ denote the vector of all the introduced extra binary variables. For each pattern $\mathbf{c}[\mathbf{s}]$, denote

$$\overline{Z_{\mathbf{c}[\mathbf{s}]}} = \sum_{v[s] \in \mathbf{c}[\mathbf{s}]} Z_{v[s]}.$$

The scoring function of the second slave is

$$h(Z, \mathbf{u}) = h_1(Z) + h_2(Z, \mathbf{u}) + h_3(Z, \mathbf{u}) + h_4(Z, \mathbf{u})$$

where

$$h_1(Z) = \sum_{v,s} \lambda_{v[s]} Z_{v[s]} + \sum_{\substack{\mathbf{c} \notin \mathcal{T} \\ |\mathbf{c}|=2}} \sum_{\mathbf{s} \in \mathbf{c}[\cdot]} \left(\psi_{\mathbf{c}} + \phi_{\mathbf{c}[\mathbf{s}]} + \sum_{\substack{\mathbf{c}'[\mathbf{s}'] \supseteq \mathbf{c}[\mathbf{s}] \\ |\mathbf{c}'|\geq 3, \phi_{\mathbf{c}'[\mathbf{s}']}<0}} \phi_{\mathbf{c}'[\mathbf{s}']} \right) Z_{\mathbf{c}[\mathbf{s}]}$$

$$h_2(Z, \mathbf{u}) = \sum_{\substack{\mathbf{c} \\ |\mathbf{c}|\geq 3}} \sum_{\substack{\mathbf{s} \in \mathbf{c}[\cdot] \\ \phi_{\mathbf{c}[\mathbf{s}]} \geq 0}} \phi_{\mathbf{c}[\mathbf{s}]} \left(\overline{Z_{\mathbf{c}[\mathbf{s}]}} - |\mathbf{c}| + 1 \right) u_{\mathbf{c}[\mathbf{s}]}$$

$$h_3(Z, \mathbf{u}) = \sum_{\substack{\mathbf{c} \\ |\mathbf{c}|=3}} \sum_{\substack{\mathbf{s} \in \mathbf{c}[\cdot] \\ \phi_{\mathbf{c}[\mathbf{s}]} < 0}} \left| \phi_{\mathbf{c}[\mathbf{s}]} \right| u_{\mathbf{c}[\mathbf{s}]} \left(\overline{Z_{\mathbf{c}[\mathbf{s}]}} - 1 \right)$$

$$h_4(Z, \mathbf{u}) = \sum_{\substack{\mathbf{c} \\ |\mathbf{c}|\geq 4}} \sum_{\substack{\mathbf{s} \in \mathbf{c}[\cdot] \\ \phi_{\mathbf{c}[\mathbf{s}]} < 0}} |\phi_{\mathbf{c}[\mathbf{s}]}| \left(u^0_{\mathbf{c}[\mathbf{s}]} (2\overline{Z_{\mathbf{c}[\mathbf{s}]}} - 3) + \sum_{j=1}^{|\mathbf{c}|-4} u^j_{\mathbf{c}[\mathbf{s}]} (\overline{Z_{\mathbf{c}[\mathbf{s}]}} - j - 2) \right).$$

Term	*Number of Variables*
$h_1(Z)$	$N^2\|\mathcal{S}\|^2$
$h_2(Z, \mathbf{u})$	$\sum_{\substack{\mathbf{c} \\ \|\mathbf{c}\|\geq 3}} \sum_{\substack{\mathbf{s} \in \mathbf{c}[\cdot] \\ \phi_{\mathbf{c}[\mathbf{s}]} \geq 0}} (1 + \|\mathbf{c}\|)$
$h_3(Z, \mathbf{u})$	$\sum_{\substack{\mathbf{c} \\ \|\mathbf{c}\|=3}} \sum_{\substack{\mathbf{s} \in \mathbf{c}[\cdot] \\ \phi_{\mathbf{c}[\mathbf{s}]} < 0}} (1 + \|\mathbf{c}\|)$
$h_4(Z, \mathbf{u})$	$\sum_{\substack{\mathbf{c} \\ \|\mathbf{c}\|\geq 4}} \sum_{\substack{\mathbf{s} \in \mathbf{c}[\cdot] \\ \phi_{\mathbf{c}[\mathbf{s}]} < 0}} (2\|\mathbf{c}\| - 3)$

Table 1: Number of variables in each part of $h(Z, \mathbf{u})$

h_1 involves the edges that are not in $\mathcal{T}$, h_2 involves positive terms of degree 3 or more. h_3 involves negative cubic terms, h_4 involves negative terms of degree 4 or more.

The relaxed problem for generalized higher order CRFs, i.e., Problem (2) is

$$\begin{aligned} \min_{\lambda} \max_{Y,Z,\mathbf{u}} \quad & f_\lambda(Y) + h(Z, \mathbf{u}) \\ \text{s.t.} \quad & Y \in \mathcal{Y} \\ & Z, \mathbf{u} \text{ are binary} \end{aligned} \tag{6}$$

3.4 Complexity Analysis

In this section, we theoretically analyze the time complexity for each iteration in dual decomposition. Running time for $\max_{Y \in \mathcal{Y}} f_\lambda(Y)$ is linear in the size of the graph, i.e., $N \times |\mathcal{S}|^2$. Running time for $\max_{Z,\mathbf{u}} h(Z, \mathbf{u})$ is cubic in the number of variables, which is the sum of variables in function h_1 to h_4. $h_1(Z)$ has at most $N^2|\mathcal{S}|^2$ variables; each pattern in $h_2(Z, \mathbf{u})$ requires one extra variable, hence $h_2(Z, \mathbf{u})$ has $\sum_{\substack{\mathbf{c} \\ |\mathbf{c}|\geq 3}} \sum_{\substack{\mathbf{s} \in \mathbf{c}[\cdot] \\ \phi_{\mathbf{c}[\mathbf{s}]} \geq 0}} (1 + |\mathbf{c}|)$ variables. Similarly, we could count the number of variables in h_3 and h_4, as shown in Table 1.

In summary, each pattern in $h(Z, \mathbf{u})$ requires at most $2|\mathbf{c}| - 2$ variables, so $h(Z, \mathbf{u})$ has no more than $\sum_{\mathbf{c}} \sum_{\mathbf{s} \in \mathbf{c}[\cdot]} (2|\mathbf{c}| - 2)$ variables.

Finally, the time complexity for each iteration in dual decomposition is

$$O\left(N|\mathcal{S}|^2 + \left(\sum_{\mathbf{c}} \sum_{\mathbf{s} \in \mathbf{c}[\cdot]} (2|\mathbf{c}| - 2) \right)^3 \right)$$

which is cubic in the total length of patterns.

4 Experimental Results

4.1 Named Entity Recognition in Tweets

4.1.1 Data Sets

Our first experiment is named entity recognition in tweets. Recently, information extraction on Twitter or Facebook data is attracting much attention (Ritter et al., 2011). Different from traditional information extraction for news articles, messages posted on these social media websites are short and noisy, making the task more challenging. In this paper, we use generalized higher order CRFs for Twitter NER with discriminative training, and compare our 2-slave dual decomposition approach with spanning tree based dual decomposition approach and other decoding algorithms.

So far as we know, there are two publicly available data sets for Twitter NER. One is the Ritter's (Ritter et al., 2011), the other is from MSM2013 Concept Extraction Challenge (Basave et al., 2013)[1]. Note that in Ritter's work (Ritter et al., 2011), all of the data are used for evaluating named entity type classification, and not used during training. However, our approach requires discriminative training, which makes our method not comparable with their results. Therefore we choose the MSM2013 dataset in our experiment and compare our system with the MSM2013 official runs.

The MSM2013 corpus has 4 types of named entities, person (PER), location (LOC), organization (ORG), and miscellaneous (MISC). The name entities are about film/movie, entertainment award event, political event, programming language, sporting event and TV show. The data is separated into a training set containing 2815 tweets, and a test set containing 1526 tweets.

4.1.2 Local Features

We cast the NER task as a structured classification problem, and adopt BIESO labeling, where for each multi-word entity of class *C*, the first word is labeled as *B-C*, the words in the entity are labeled as *I-C*, and the last word is labeled as *E-C*, a single word entity of class *C* is labeled as *S-C*, and other words are labeled as *O*.

[1]http://oak.dcs.shef.ac.uk/msm2013/challenge.html

Our baseline NER is a linear chain CRF. As the MSM2013 competition allows to use extra resources, we use several additional datasets to generate rich features. Specifically, we trained two POS taggers and two NER taggers using extra datasets. All the 4 taggers are trained using linear chain CRFs with perceptron training. One POS tagger is trained on Brown and Wall Street Journal corpora in Penn Tree Bank 3, and the other is trained on ARK Twitter NLP corpus (Gimpel et al., 2011) with slight modification. One of the NER taggers is trained on CoNLL 2003 English dataset[2], and the other is trained on Ritter's dataset.

We used dictionaries in Ark Twitter NLP toolkit[3], Ritter's Twitter NLP toolkit[4] and Moby Words project[5] to generate dictionary features. We also collected film names and TV shows from IMDB website and musician groups from wikipedia. These dictionaries are used to detect candidate named entities in the training and testing datasets using string matching. Those matched words are assigned with BIESO style labels which are used as features.

We also used the unsupervised word cluster features provided by Ark Twitter NLP toolkit, which has significantly improved the Twitter POS tagging accuracy (Owoputi et al., 2013). Similar with previous work, we used prefixes of the cluster bit strings with lengths $\in \{2, 4, \ldots, 16\}$ as features.

4.1.3 Global Features

Previous studies showed that the document level consistency features (same phrases in a document tend to have the same entity class) are effective for NER (Kazama and Torisawa, 2007; Finkel et al., 2005). However, unlike news articles, tweets are not organized in documents. To use these document level consistency features, we grouped the tweets in MSM2013 dataset using single linkage clustering algorithm where similarity between two tweets is the number of their overlapped words. If the similarity is greater than 4, then we put the two tweets into one group. Unlike standard document clustering, we did not normalize the length of tweets since all the tweets are limited to 140 characters. Then we

[2]www.cnts.ua.ac.be/conll2003/

[3]https://code.google.com/p/ark-tweet-nlp/

[4]http://github.com/aritter/Twitter_nlp

[5]http://icon.shef.ac.uk/Moby/mwords.html

extracted the group level features as follows. For any two identical phrases $x_i \dots x_{i+k}$, $x_j \dots x_{j+k}$ in a group, a binary feature is true if they have the same label subsequences. The pattern set of this feature is $\mathbf{c} = \{i, \dots, i+k, j, \dots, j+k\}$ and $\mathbf{c}[\cdot] = \{\mathbf{s} | s_i = s_j, \dots, s_{i+k} = s_{j+k}\}$.

4.1.4 Results

We use two evaluation metrics. One is the micro averaged F score, which is used in CoNLL2003 shared task. The other is macro averaged F score, which is used in MSM2013 official evaluation (Basave et al., 2013).

We compare our approach with two baselines, integer linear programming (ILP)[6] and a naive dual decomposition method. In naive dual decomposition, we use three types of slaves: a linear chain captures unigrams and bigrams, and the spanning trees cover the skip edges linking identical words. Identical multi-word phrases yield larger factors with more than 4 vertices. They could not be handled efficiently by belief propagation for spanning trees. Therefore, we create multiple slaves, each of which covers a pair of identical multi-word phrases.

To reduce the number of slaves, we use a greedy algorithm to choose the spanning trees. Each time we select the spanning tree that covers the most uncovered edges. This can be done by performing the maximum spanning tree algorithm on the graph where each uncovered edge has unit weight. Let x^* denote the most frequent word in a tweet cluster, and F^* is its frequency, then at least $(F^*-1)/2$ spanning trees are required to cover the complete subgraph spanned by x^*.

For both dual decomposition systems, averaged perceptron (Collins, 2002a) with 10 iterations is used for parameter estimation. We follow the work of Rush et al. (2010) to choose the step size in the sub-gradient algorithm.

Table 2 shows the comparison results, including two F scores and total running time (seconds) for training and testing. Performances of the top 4 official runs are also listed. Different from our approach, the top performing systems mainly benefit from rich open resources, such as DBpedia Gazetteer, ANNIE Gazetteer, Yago, Microsoft N-grams, and external NER system combination, such as ANNIE, OpenNLP, LingPipe, OpenCalais (Basave et al., 2013). We can see that general CRFs with global features are competitive with these top systems. Our 2-slave DD outperforms naive DD and achieves competitive performance with exact inference based on ILP, while is much faster than ILP.

[6] we use Gurobi as the ILP solver, http://www.gurobi.com/

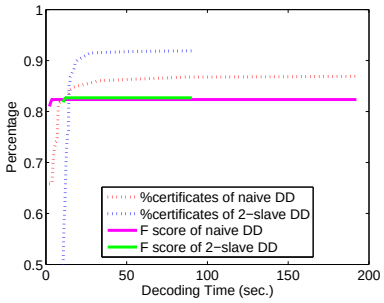
System	F_{macro}	F_{micro}	Sec.
Linear chain CRFs	0.657	0.815	98
General CRFs (2-slave DD)	**0.680**	**0.827**	214
General CRFs (naive DD)	0.672	0.824	490
General CRFs (ILP)	**0.680**	**0.828**	8640
Official 1st	0.670	N/A	N/A
Official 2nd	0.662	N/A	N/A
Official 3rd	0.658	N/A	N/A
Official 4th	0.610	N/A	N/A

Table 2: Comparison results on MSM2013 Twitter NER task.

To compare the convergence speed and optimality of 2-slave DD and naive DD algorithms, we use the model trained by ILP, and record the F_{micro} scores, averaged dual objectives per instance (the lower the tighter), decoding time, and fraction of optimality certificates across iterations of the two DD algorithms on test data. Figure 2 shows the performances of the two algorithms relative to decoding time. Our method requires 0.0064 seconds for each iteration on average, about four times slower than the naive DD. However, our approach achieves a tighter upper bound and larger fraction of optimality certificates.

4.2 Sentence Dependency Tagging

Our second experiment is sentence dependency tagging in Question Answering forums task studied in Qu and Liu's work (Qu and Liu, 2012). The goal is to extract the dependency relationships between sentences for automatic question answering. For example, from the posts below, we would need to know that sentence $S4$ is a comment about sentence $S1$ and $S2$, not an answer to $S3$.



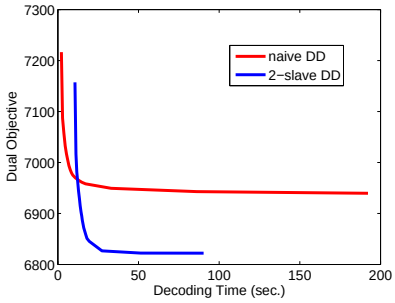


Figure 2: Twitter NER: The F_{micro} scores, dual objectives, and fraction of optimality certificates relative to decoding time.

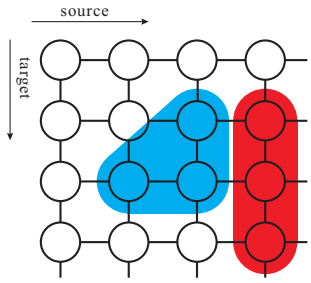


Figure 3: 3-wise CRF for QA sentence dependency tagging. Order-3 factors (e.g., red and blue) connects the 3 vertices in adjacent edge pairs.

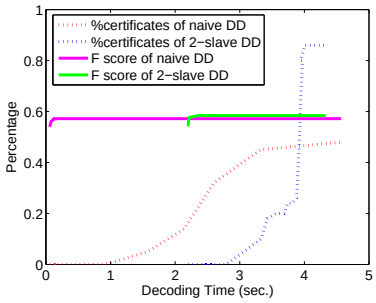
A: **[S1]**I'm having trouble installing my DVB Card.
[S2]dmesg prints: . . .
[S3]What could I do to resolve this problem?
B: **[S4]** I'm having similar problems with Ubuntu

For a pair of sentences, the depending sentence is called the source sentence, and the depended sentence the target sentence. One source sentence can potentially depend on many different target sentences, and one target sentence can also correspond to multiple sources. Qu and Liu (2012) casted the task as a binary classification problem, i.e., whether or not there exists a dependency relation between a pair of sentences. Formally, in this task, Y is a $N^2 \times 2$ matrix, where N is the number of sentences, $Y_{i*N+j[1]} = 1$ if the i^{th} sentence depends on the j^{th} sentence, otherwise, $Y_{i*N+j[0]} = 1$. We use the corpus in Qu and Liu's work (Qu and Liu, 2012), where dependencies between $3,483$ sentences in 200 threads were annotated. Following their settings we randomly split annotated threads into three disjoint sets, and run a three-fold cross validation. F score is used as the evaluation metric.

System	F	Sec.
2D CRFs (naive)	0.564	9.2
2D CRFs (2-slave)	0.565	16.4
3-wise CRFs (naive)	0.572	18.7
3-wise CRFs (2-slave)	0.584	17.33
(Qu and Liu, 2012)	0.561	N/A

Table 3: Comparison results on QA sentence dependency tagging task.

Qu and Liu (2012) used the pairwise CRF with a 4-connected neighborhood system (2D CRF) as their graphical model, where each vertex in the graph represents a sentence pair, and each edge connects adjacent source sentences or target sentences. The key observation is that given a source/target sentence, there is strong dependency between adjacent target/source sentences. In this paper, we extend their work by connecting the 3 vertices in adjacent edge pairs, resulting in 3-wise CRFs, as shown in Figure 3. We use the same vertex features and edge features as in Qu and Liu's work. For a 3-tuple of vertices, we use the following features: combination of the sentence types within the tuple, whether the related sentences are in one post or belong to the same author. Again, we use perceptron to train the model, and the max iteration number for dual decomposition is 200. The spanning tree in our decomposition is the concatenation of all the rows in the graph.



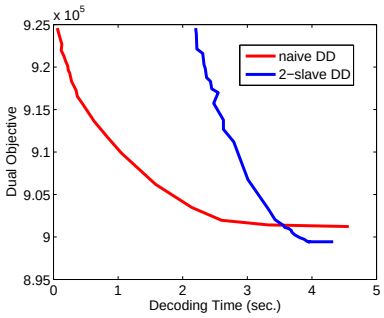


Figure 4: QA sentence dependency tagging using 3-wise CRFs: The F scores, dual objectives, and fraction of optimality certificates relative to decoding time.

Table 3 shows the experimental results. For 2D CRFs, the edges can be covered by 2 spanning trees (one covers all vertical edges and the other covers all horizontal edges), hence the naive dual decomposition has only two slaves. Compared with naive DD, our 2-slave DD achieved competitive performance while two times slower. This is because naive DD adopts dynamic programming that runs in linear time. However, for 3-wise CRFs, the naive dual decomposition requires many small slaves to cover the order-3 factors. Therefore our 2-slave method is more effective. The fraction of optimality certificates and dual objectives of 3-wise CRFs relative to decoding time during testing are shown in Figure 4. For each iteration, our method requires 0.0049 seconds and the naive DD requires 0.00054 seconds, about 10 times faster than ours, but our method converges to a lower lower bound.

5 Conclusion

We proposed a new decomposition approach for generalized higher order CRFs using only two slaves. Both permit polynomial decoding time. We evaluated our method on two different tasks: Twitter named entity recognition and forum sentence dependency detection. Experimental results show that though the compact decomposition requires more running time for each iteration, it achieves consistently tighter bounds and outperforms the naive dual decomposition. The two experiments demonstrate that our method works for general graphs, even if the graph can not be decomposed into a few spanning trees (for example, if the graph has large complete subgraphs or large factors).

Our code is available at
https://github.com/qxred/higher-order-crf

Appendix A

We show that by setting a sufficiently large ψ, $g_\lambda(Z)$ in Section 3.2 is graph representable.

Let

$$g_\lambda(Z) = g_1(Z) + g_2(Z) + g_3(Z) + g_4(Z)$$

where

$$\begin{aligned}
g_1(Z) &= \sum_{\substack{\mathbf{c}\\|\mathbf{c}|=2}} \sum_{\mathbf{s}\in\mathbf{c}[\cdot]} \left(\psi_{\mathbf{c}} + \phi_{\mathbf{c}[\mathbf{s}]}\right) Z_{\mathbf{c}[\mathbf{s}]} + \sum_{v,s} \lambda_{v[s]} Z_{v[s]} \\
g_2(Z) &= \sum_{\substack{\mathbf{c}\\|\mathbf{c}|\geq 3}} \sum_{\substack{\mathbf{s}\in\mathbf{c}[\cdot]\\ \phi_{\mathbf{c}[\mathbf{s}]}\geq 0}} \phi_{\mathbf{c}[\mathbf{s}]} Z_{\mathbf{c}[\mathbf{s}]} \\
g_3(Z) &= \sum_{\substack{\mathbf{c}\\|\mathbf{c}|=3}} \sum_{\substack{\mathbf{s}\in\mathbf{c}[\cdot]\\ \phi_{\mathbf{c}[\mathbf{s}]}< 0}} \phi_{\mathbf{c}[\mathbf{s}]} Z_{\mathbf{c}[\mathbf{s}]} \\
g_4(Z) &= \sum_{\substack{\mathbf{c}\\|\mathbf{c}|\geq 4}} \sum_{\substack{\mathbf{s}\in\mathbf{c}[\cdot]\\ \phi_{\mathbf{c}[\mathbf{s}]}< 0}} \phi_{\mathbf{c}[\mathbf{s}]} Z_{\mathbf{c}[\mathbf{s}]}
\end{aligned}$$

For $g_2(Z)$, since coefficients of all terms are non-negative, we can use the fact

$$\prod_{\substack{i\\ a_i\in\{0,1\}}} a_i = \max_{b\in\{0,1\}} \left(\sum_i a_i - |\mathbf{a}| + 1\right) b \qquad (7)$$

to reduce $g_2(Z)$ into an equivalent quadratic form (Freedman and Drineas, 2005). That is, $\max_Z g_2(Z)$ is equivalent to

$$\max_{Z,\mathbf{u}} \sum_{\substack{\mathbf{c},|\mathbf{c}|\geq 3\\ \phi_{\mathbf{c}[\mathbf{s}]}\geq 0}} \sum_{\mathbf{s}\in\mathbf{c}[\cdot]} \phi_{\mathbf{c}[\mathbf{s}]} \left(\sum_{v[s]\in\mathbf{c}[\mathbf{s}]} Z_{v[s]} - |\mathbf{c}| + 1\right) u_{\mathbf{c}[\mathbf{s}]}$$

which is graph representable because coefficients of all the quadratic terms are non-negative.

Coefficients of terms in $g_3(Z)$ and $g_4(Z)$ are negative, therefore $g_3(Z)$ and $g_4(Z)$ are not supermodular. To make them graph representable, we use the following fact

Proposition 1 *(Živný and Jeavons, 2010) The pseudo-Boolean function* $p(\mathbf{x}) = \sum_{1\leq i<j\leq K} x_i x_j - \prod_{i=1}^{K} x_i$ *is graph representable and can be reduced to the quadratic forms: if* $K = 3$*, then*

$$p(\mathbf{x}) = \max_{y\in\{0,1\}} (x_1 + x_2 + x_3 - 1)y \tag{8}$$

otherwise $K > 3$*,*

$$\begin{aligned} p(\mathbf{x}) &= \max_{y_0\in\{0,1\}} y_0(2\sum_{i=1}^{K} x_i - 3) \\ &+ \max_{\text{binary } \mathbf{y}} \sum_{j=1}^{K-4} y_j(\sum_{i=1}^{K} x_i - j - 2) \end{aligned} \tag{9}$$

According to Eq (8), for each cubic term in $g_3(Z)$, we have

$$\begin{aligned} & \phi_{\mathbf{c}[\mathbf{s}]} \prod_{v[s]\in\mathbf{c}[\mathbf{s}]} Z_{v[s]} \\ = & \left|\phi_{\mathbf{c}[\mathbf{s}]}\right| \left(\sum_{u[s],v[t]\in\mathbf{c}[\mathbf{s}]} Z_{u[s]}Z_{v[t]} - \prod_{v[s]\in\mathbf{c}[\mathbf{s}]} Z_{v[s]} \right) \\ & - \left|\phi_{\mathbf{c}[\mathbf{s}]}\right| \sum_{u[s],v[t]\in\mathbf{c}[\mathbf{s}]} Z_{u[s]}Z_{v[t]} \\ = & \left|\phi_{\mathbf{c}[\mathbf{s}]}\right| \max_{u_{\mathbf{c}[\mathbf{s}]}\in\{0,1\}} u_{\mathbf{c}[\mathbf{s}]} \left(\sum_{v[s]\in\mathbf{c}[\mathbf{s}]} Z_{v[s]} - 1 \right) \\ & - \left|\phi_{\mathbf{c}[\mathbf{s}]}\right| \sum_{u[s],v[t]\in\mathbf{c}[\mathbf{s}]} Z_{u[s]}Z_{v[t]} \end{aligned}$$

The first part on the right hand side is graph representable since all quadratic terms are non-negative. The second part is a quadratic function of Z, and it can be merged into $g_1(Z)$. With sufficiently large $\psi_{\mathbf{c}}$ in $g_1(Z)$, we could guarantee the non-negativity of all quadratic terms.

Similarly, we could apply Eq (9) to reduce $g_4(Z)$ to graph representable quadratic forms.

Acknowledgments

We thank three anonymous reviewers for their valuable comments. This work is partly supported by DARPA under Contract No. FA8750-13-2-0041. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Amparo Elizabeth Cano Basave, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. 2013. Making sense of microposts (msm2013) concept extraction challenge (challenge report). In *Proceedings of the Concept Extraction Challenge at the Workshop on 'Making Sense of Microposts'*, pages 1–15.

A. Billionnet and M. Minoux. 1985. Maximizing a supermodular pseudoboolean function: A polynomial algorithm for supermodular cubic functions. *Discrete Applied Mathematics*, 12(1):1 – 11.

Endre Boros and PeterL. Hammer. 1991. The max-cut problem and quadratic 0-1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33(3):151–180.

Endre Boros and Peter L. Hammer. 2002. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1C3):155 – 225.

Michael Collins. 2002a. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.

Michael Collins. 2002b. Ranking algorithms for named entity extraction: Boosting and the votedperceptron. In *Proceedings of ACL*, pages 489–496, July.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. 2008. Using conditional random fields to extract contexts and answers of questions from online forums. In *Proceedings of ACL-08: HLT*, pages 710–718, June.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370, June.

Daniel Freedman and Petros Drineas. 2005. Energy minimization via graph cuts: Settling what is possible. In *Proceedings of CVPR*, pages 939–946. IEEE Computer Society.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of EMNLP*, pages 364–372.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: annotation, features, and experiments. In *Proceedings of ACL-HLT*, HLT '11, pages 42–47.

AndrewV. Goldberg. 2008. The partial augmentcrelabel algorithm for the maximum flow problem. In Dan Halperin and Kurt Mehlhorn, editors, *Algorithms - ESA 2008*, volume 5193 of *Lecture Notes in Computer Science*, pages 466–477. Springer Berlin Heidelberg.

Vladimir Jojic, Stephen Gould, and Daphne Koller. 2010. Accelerated dual decomposition for map inference. In *Proceedings of ICML*, pages 503–510. Omnipress.

Jun'ichi Kazama and Kentaro Torisawa. 2007. A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of EMNLP-CoNLL*, pages 315–324, June.

Vladimir Kolmogorov and Ramin Zabih. 2004. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159.

Vladimir Kolmogorov. 2006. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, October.

Nikos Komodakis. 2011. Efficient training for pairwise or higher order CRFs via dual decomposition. In *Proceedings of CVPR*, pages 1841–1848. IEEE Computer Society.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298, Cambridge, MA, October.

Philipp Krähenbühl and Vladlen Koltun. 2011. Efficient inference in fully connected crfs with gaussian edge potentials. In *Proceedings of NIPS*, pages 109–117.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Andre Martins, Mario Figueiredo, Pedro Aguiar, Noah Smith, and Eric Xing. 2011a. An augmented lagrangian approach to constrained map inference. In *Proceedings of ICML*, pages 169–176. ACM, June.

Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. 2011b. Dual decomposition with many overlapping components. In *Proceedings of the EMNLP*, pages 238–249, July.

G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1):265–294.

James B. Orlin. 2009. A faster strongly polynomial time algorithm for submodular function minimization. *Math. Program.*, 118(2):237–251.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*, pages 380–390, June.

Xian Qian, Xiaoqian Jiang, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Sparse higher order conditional random fields for improved sequence labeling. In *Proceedings of ICML*, volume 382, page 107. ACM.

Zhonghua Qu and Yang Liu. 2012. Sentence dependency tagging in online question answering forums. In *Proceedings of ACL*, pages 554–562, July.

J. M. W. Rhys. 1970. A selection problem of shared fixed costs and network flows. *Management Science*, 17(3):200–207.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of EMNLP*, pages 1524–1534, July.

Alexander M. Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *J. Artif. Int. Res.*, 45(1):305–362, September.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP 2010*, pages 1–11, Cambridge, MA, October.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proceedings of NIPS*.

Charles Sutton and Andrew Mccallum, 2006. *Introduction to Conditional Random Fields for Relational Learning*. MIT Press.

Rustem Takhanov and Vladimir Kolmogorov. 2013. Inference algorithms for pattern-based CRFs on sequence data. In *Proceedings of ICML*, pages 145–153.

Stanislav Živný and Peter G. Jeavons. 2010. Classes of submodular constraints expressible by graph cuts. *Constraints*, 15(3):430–452, July.

Nan Ye, Wee Sun Lee, Hai Leong Chieu, and Dan Wu. 2009. Conditional random fields with high-order features for sequence labeling. In *Proceedings of NIPS*, pages 2196–2204.