Journal of Machine Learning Research 11 (2010) 3097-3135 Submitted 7/09; Revised 6/10; Published 11/10

Collective Inference for Extraction MRFs Coupled with Symmetric Clique Potentials

Rahul Gupta GRAHUL@CSE.IITB.AC.IN
Sunita Sarawagi SUNITA@IITB.AC.IN
Ajit A. Diwan AAD@CSE.IITB.AC.IN
Computer Science and Engineering
IIT Bombay, Powai
Mumbai, 400076, India

Editor: Ben Taskar

Abstract

Many structured information extraction tasks employ collective graphical models that capture inter-instance associativity by coupling them with various clique potentials. We propose tractable families of such potentials that are invariant under permutations of their arguments, and call them *symmetric clique potentials*. We present three families of symmetric potentials—MAX, SUM, and MAJORITY.

We propose cluster message passing for collective inference with symmetric clique potentials, and present message computation algorithms tailored to such potentials. Our first message computation algorithm, called α-pass, is sub-quadratic in the clique size, outputs exact messages for MAX, and computes $\frac{13}{15}$-approximate messages for Potts, a popular member of the SUM family. Empirically, it is upto two orders of magnitude faster than existing algorithms based on graph-cuts or belief propagation. Our second algorithm, based on Lagrangian relaxation, operates on MAJORITY potentials and provides close to exact solutions while being two orders of magnitude faster. We show that the cluster message passing framework is more principled, accurate and converges faster than competing approaches.

We extend our collective inference framework to exploit associativity of more general *intra-domain properties* of instance labelings, which opens up interesting applications in domain adaptation. Our approach leads to significant error reduction on unseen domains without incurring any overhead of model retraining.

Keywords: graphical models, collective inference, clique potentials, cluster graphs, message passing

1. Introduction

Markov Random Fields (MRFs) are the models of choice in various structured information extraction (IE) tasks such as part-of-speech tagging, NP-chunking, text segmentation, and named entity recognition. The goal of IE is to mark each token in a sentence with a label from a discrete set, like Person, Location, and Other. As illustrated in Figure 1(c), the basic MRF extraction model defines edge potentials between labels of adjacent words to capture first-order dependencies (Lafferty et al., 2001). The resulting chain model allows tractable exact inference that is, computing the maximum a-posteriori (MAP) labeling of the sentence is easy.

Apart from traditional settings, IE is now being increasingly used in many web scenarios, such as query-driven extraction (Gupta and Sarawagi, 2009; Carlson et al., 2010) and constructing on-the-fly relations from semi-structured web sources (Elmeleegy et al., 2009). However these setups are characterized by limited training data, which restricts the robustness of the resulting MRFs during deployment. One promising way to increase the robustness of MRFs during deployment is to strengthen the first-order dependencies of the model with extra long-range dependencies , resulting in more robust labelings. The goal of this paper is to exploit long-range dependencies in a principled manner.

There has been a lot of work on capturing long-range dependencies between labels of non-adjacent words. This typically includes dependencies of the form—if a token repeats in a document, then the labels of the repetitions should ideally be the same (Sutton and McCallum, 2004; Bunescu and Mooney, 2004; Krishnan and Manning, 2006; Finkel et al., 2005). In Figure 1(d), we show an example where the extra dotted edges join unigram recurrences. Such long-range dependencies are termed *associative* since they introduce positive couplings between labels of token pairs. Apart from textual tokens, associative dependencies have also been exploited while labeling documents (Lu and Getoor, 2003; Chakrabarti et al., 1998), image pixels (Taskar et al., 2004), and annotating Web documents (Kulkarni et al., 2009). Due to these couplings, MAP inference is done collectively across all sentences so as to maximize the sum of sentence-specific and long-range associative potentials. This joint MAP computation task is traditionally called **collective inference**.

Previous work on collective inference can be broadly classified into two categories. The first category defines a separate associative potential over every inter-instance edge (i.e., dotted edge in Figure 1(d)). Inference on such graphs is performed by viewing it as a graphical model with pairwise potentials. A variety of generic approximation algorithms have been used to solve this typically intractable inference task, including Loopy Belief propagation (Bunescu and Mooney, 2004; Sutton and McCallum, 2004), and Gibbs sampling (Finkel et al., 2005). The second category defines an associative potential for each clique that is created from all repetitions of a unigram and the potentials can take more general forms like the Majority function (Krishnan and Manning, 2006). Collective inference on this category of models is performed using local search algorithms such as Iterative Conditional Mode fitting (ICM) (Lu and Getoor, 2003; Chakrabarti et al., 1998) or two stage algorithms (Krishnan and Manning, 2006).

This paper unifies various collective extraction tasks with different forms of associative potentials under a single framework. We do this by employing a cluster graph representation of the collective model. Figure 1(e) illustrates the cluster graph for our toy example. The cluster graph comprises of one cluster per MRF-instance, and one cluster per clique with its corresponding associative potential. As in the traditional collective extraction models, we assume that a clique comprises of all the occurrences of a particular token. We emphasize that instead of clusters for chain models, the framework can also define clusters for any tractable component, such as a bipartite matching or an alignment, but since our interest is in IE tasks, we shall focus on chain models.

Collective inference in our model then simply corresponds to message passing in this cluster graph. This view of collective inference offers several advantages over earlier approaches. First, it allows us to plug in and study various clique potentials cleanly under the same cluster message passing umbrella. Second, it allows us to exploit special properties of the associative potentials to design combinatorial algorithms that are both more accurate and more efficient than existing algorithms. Specifically, we show that most associative potentials used in practice are **symmetric clique potentials**. Symmetric clique potentials are invariant under any permutation of their arguments. So the value of a symmetric potential depends only on the counts of its distinct arguments, and not their position in the clique. For example in Figure 1(d), the clique potential on 'Iraq' would give a low score if its end vertices had different labels, regardless of which 'Iraq' vertex gets what label. We present three families of symmetric clique potentials that capture label associativity in different ways—MAX, SUM (which subsumes the popular Potts potential), and MAJORITY. The most crucial component of the collective inference problem is then to efficiently compute outgoing messages from clique clusters governed by symmetric potentials. We denote this sub-problem as **clique inference**.

1.1 Contributions

We show that the overall collective inference problem is intractable even for the case of two labels. We therefore concentrate on designing efficient and accurate message passing algorithms on our special cluster graph. We present a suite of efficient combinatorial algorithms tailored to specific symmetric clique potentials for the clique inference sub-problem. We devised a combinatorial algorithm called α-pass that computes exact outgoing messages for cliques with MAX potentials, and also for any arbitrary symmetric clique potential over two labels. We show that α-pass provides a $\frac{13}{15}$-approximation for clique inference with the well-known and NP-hard Potts potential with the SUM family. We show that this analysis is tight, and that the corresponding clique inference bound by alternative schemes like α-expansion, LP-rounding, TRW-S and ICM are either $\frac{1}{2}$. or at best locally optimal. Further, the runtime of α-pass is $O(mn\log n)$ where n is the clique size, and m is the number of labels. We also show that α-pass can be generalized to provide a better approximation of $\frac{8}{9}$, but with a runtime of $O(m^2 n \log n)$. Alternative clique inference approaches such as the graph-cut algorithm of Boykov et al. (2001) and the tree-reweighted message passing (TRWS) algorithm of Kolmogorov (2006) are quadratic in n. We present a new Lagrangian-relaxation based algorithm, called LR, for MAJORITY potentials. The LR algorithm is nearly exact in practice but is usually two orders of magnitude faster than an exact LP-based algorithm.

Our experiments show that computing a rich set of messages leads to significantly more accuracy gains over other collective inference approaches that do not compute such messages, for example, Krishnan and Manning (2006). We also show that decomposing the problem over the cluster graph and employing fast message computation algorithms helps our collective inference scheme converge one-order faster than alternatives like loopy belief propagation. In short, we show that it makes more sense to compute messages at a cluster level, and we provide fast and accurate algorithms to do so.

We then extend our collective inference framework to capture associativity of a more general kind than just labels of unigram repetitions. We encourage associativity of *properties* of labelings of records appearing as a group. We apply this framework to domain adaptation, where a model trained on one or more domains is deployed on a different but related domain. For example, a model trained for extracting fields of a bibliographic records is deployed on all records coming from the homepage of a single author. The bibliographic entries from the same author's publications homepage would predominantly have the same style, such as order of labels, say Title followed by Author(s) followed by Venue, or the HTML tag before a Title. While we do not know the property values across the records apriori, we do know that they will be largely unimodal across records within a given domain. We use this collective signal to couple together the labelings of intra-domain records, and show how our collective inference framework naturally extends to this scenario and provides significant reduction in error.

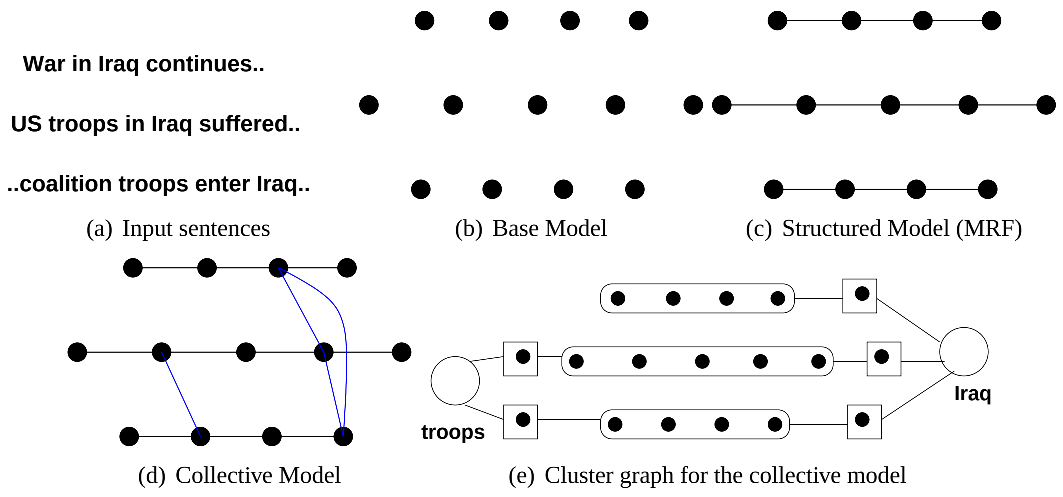

War in Iraq continues..
US troops in Iraq suffered..
..coalition troops enter Iraq..
(a) Input sentences
(b) Base Model
(c) Structured Model (MRF)
(d) Collective Model
troops
Iraq
(e) Cluster graph for the collective model

Figure 1: Various models for named-entity recognition illustrated on a small corpus. Figure 1(e) shows the cluster graph for the collective model of (d) with one cluster per sentence (shown as flat chains), and one cluster per associative clique (shown as big circles).

At this point, we note that our previous work (Gupta et al., 2007) focused primarily on the α-pass algorithm along with a sketch of some of its properties. This paper presents a detailed treatment of the algorithm and its generalized version, with rigorous proofs of their approximation bounds, including some new bounds for a variant of the clique inference objective with Potts potential. In addition, new material in this paper includes the LR algorithm for the MAJORITY potential, extension of our framework to include properties-based associativity, and empirical studies of various collective inference techniques.

We also note that subsequent to our work (Gupta et al., 2007), there has been an abundance of research on higher order clique potentials in last few years, primarily by the computer vision community. While their applications and basic graph structure are different (grids vs chains), many of their ideas are highly relevant—either in terms of special algorithms for clique/collective inference (Komodakis et al., 2007a; Komodakis and Paragios, 2009; Kumar and Torr, 2008a; Werner, 2008), reduction of clique potentials to pairwise potentials (Kohli et al., 2009; Ishikawa, 2009), and special families of clique potentials (Potetz and Lee, 2008; Rother et al., 2009; Tarlow et al., 2010). We will present theoretical and empirical comparisons and draw parallels with the relevant schemes later on in this paper.

1.2 Outline

In Section 2, we present the MRF model for extraction and define the collective inference problem in the traditional setup of unigrams. We show that even with this setup, collective inference remains NP-hard. In Section 3, we discuss the cluster message passing algorithm for collective inference, and introduce the *clique inference* sub-problem which formalizes the task of computing outbound messages from clique clusters. Then in Section 4, we describe the MAX, SUM, and MAJORITY families of symmetric potentials. In Section 5 we present the α-pass and LR clique inference algorithms, and analyze their approximation quality. In Section 6, we extend our framework to capture more general associative properties. Section 7 contains experimental results of three types: (a) quality of the α-pass and LR algorithms; (b) effect of modeling more general associative properties in a domain adaptation task; and (c) collective inference via cluster message passing vs alternatives. Finally, Section 8 discusses prior art and Section 9 contains conclusions and a discussion of future work.

2. Collective Inference with MRFs

We now review the basic first-order MRF model for information extraction, and then formally define the collective inference problem over various coupled MRFs. We denote a sentence by bold $\mathbf{x}$, and its labeling vector by $\mathbf{y}$. Consequently, x_i and y_i denote the i^{th} token in $\mathbf{x}$ and its label respectively. The MRF defines a log-linear model parameterized by a weight vector $\mathbf{w}$, where the conditional probability of a labeling is given by:

$$\log P(\mathbf{y}|\mathbf{x},\mathbf{w}) = \sum_i \phi_i(y_i, y_{i-1}; \mathbf{x},\mathbf{w}) - \log Z_{\mathbf{w}}.$$

Here $\phi_i(y_i, y_{i-1}; \mathbf{w})$ is the log of the potential on the edge between tokens i and $i-1$, These potentials are local, that is, they depend only on the labels of the edge and $\log Z_{\mathbf{w}}$ is the normalization factor. During inference, we compute the most probable labeling of $\mathbf{x}$:

$$\arg\max_{\mathbf{y}} \log P(\mathbf{y}|\mathbf{x},\mathbf{w}) = \arg\max_{\mathbf{y}} \sum_i \phi_i(y_i, y_{i-1}; \mathbf{x},\mathbf{w}).$$

If there are m possible labels at each token, and n tokens in $\mathbf{x}$, then exact inference can be done using max-product message passing in $O(nm^2)$ time.

We now move to the collective model whose toy example is illustrated in Figure 1(d). Let there be N sentences $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$. Correspondingly, we have N conditional distributions $P(\mathbf{y}^k|\mathbf{x}^k,\mathbf{w}),\ k = 1,\ldots,N$. We use the shorthand $\phi_i^k(y_i^k, y_{i-1}^k)$ to refer to $\phi_i^k(y_i^k, y_{i-1}^k; \mathbf{x}^k, \mathbf{w})$.

Let t denote a token that repeats across the sentences, and let $\mathcal{T}$ denote the set of all such tokens. For a repeating token t, let $D(t)$ be the set of all (sentence-id, token-id) locations where it occurs in the N sentences, that is, $D(t) = \{(k,i)|x_i^k = t\}$. We express the conformance in the label assigned to positions in $D(t)$ with a higher order clique potential $\mathrm{C}_t(\{y_i^k\}_{(k,i)\in D(t)})$. Our collective MRF model that couples sentences using unigrams is then given by (up to a normalization constant):

$$\begin{aligned} \log P(\{\mathbf{y}^k\}_{k=1}^N | \{\mathbf{x}^k\}_{k=1}^N, \mathbf{w}) &\approx \sum_{k=1}^N \log P(\mathbf{y}^k|\mathbf{x}^k,\mathbf{w}) + \sum_{t\in\mathcal{T}} \mathrm{C}_t(\{y_i^k\}_{(k,i)\in D(t)}) \qquad (1) \\ &\approx \left(\sum_{k=1}^N \sum_{i=1}^{|\mathbf{x}^k|} \phi_i^k(y_i^k, y_{i-1}^k) \right) + \sum_{t\in\mathcal{T}} \mathrm{C}_t(\{y_i^k\}_{(k,i)\in D(t)}). \end{aligned}$$

The clique potential C_t has two important properties: First, it is invariant under any permutation of its arguments—that is, it is a *symmetric* function of its input. Thus, if there are m possible labels, we can represent the arguments with an m-bin histogram. For example, if $m = 3$ and input is $\{1,2,1,1,2,3,2\}$, then $\mathrm{C}_t(S)$ depends on the 3-bin histogram with values $3,3,1$. Second, C_t is *associative*—that is, it favors agreement of labels in its argument set, and it maximized when all arguments have the same label.

All collective extraction methods proposed in the literature fit the above framework. The earliest and the most popular collective model used associative potentials over every pair of occurrences of t (Sutton and McCallum, 2004; Finkel et al., 2005). This pairwise potential was a Potts potential—1 if the pair had matching labels and 0 otherwise. Clearly, we can define an appropriate C_t that mimics this behavior while staying symmetric. Consider the clique potential:

$$\mathrm{C}_t(\{y_i^k|(k,i) \in D(t)\}) = \sum_{(k,i),\ (l,j)\in D(t)} \delta(y_i^k = y_j^l).$$

This is equivalent to all the pairwise potentials for t. Also, using the histogram view, this can be re-written up to a constant as $\sum_{\alpha=1}^{m} n_\alpha^2$, where n_α is the bin count for label α in the histogram corresponding to $\{y_i^k|(k,i) \in D(t)\}$. Other known collective models like the ones proposed by Lu and Getoor (2003) and Krishnan and Manning (2006) can also be cast using the 'Majority' symmetric clique potential as we shall see in later sections.

Having defined our collective model, and shown that it subsumes popular collective models, we are ready to define the collective inference objective.

Definition 1 (Collective Inference) *Collective inference is the task of finding labelings* $\mathbf{y}^1,\ldots,\mathbf{y}^N$ *that jointly maximize the probability of the collective model (Equation 1):*

$$\arg\max_{\mathbf{y}^1,\ldots,\mathbf{y}^N} \left(\sum_{k=1}^{N} \sum_{i=1}^{|\mathbf{x}^k|} \phi_i^k(y_i^k, y_{i-1}^k) \right) + \sum_{t\in\mathcal{T}} C_t(\{y_i^k\}_{(k,i)\in D(t)}). \qquad (2)$$

In general terms, collective inference corresponds to labeling the nodes of a generic cyclic graph. However, our graph has a special structure—it is composed of chains and cliques, and although the intra-chain edge potentials ϕ can be arbitrary, the clique potentials are associative and symmetric. However we next prove that even in this setup, collective inference is NP-hard.

Theorem 2 *The collective inference problem is NP-hard even with just two labels.*

Proof We reduce the well-known NP-hard problem of finding MAX-CUTs in an arbitrary graph G to our collective inference problem (called CI). For each node $u \in G$ define a 2-node chain $u_1\ u_2$ in CI where each node can take binary labels and with edge potential $\phi^u(y,y') = M\delta(y \neq y')$. M is a large positive number $> 2E$, where E is the number of edges in G. For an edge $(u,v) \in G$, we add three cliques: $(u_1,v_1),(u_1,v_2)$ and (u_2,v_1). All three clique potentials are $\mathrm{C}_t(\{y,y'\}) = \delta(y = y')$.

The key observation is that G has a cut of size $\geq k$ iff CI has a map score of $\geq Mn+E+k$ where n is the number of nodes in G.

Suppose G has a cut (A,B) of size $\geq k$. Define a labeling in CI where all variables in $A' \triangleq \{v_1|v \in A\} \cup \{v_2|v \in B\}$ are labeled 0 and the rest B' are labeled 1. Then for every $u \in G$, (u_1,u_2) are labeled differently, thus the total contribution from the edge potentials within chains of CI is Mn. For an edge $(u,v) \in G$ that is part of the cut (A,B) with $u \in A, v \in B$, two cliques (u_1,v_2) and (u_2,v_1) have both their arguments labeled the same whereas for all other edges only one of the cliques (u_1,v_1) reaches such conformance. Thus, the total contribution from clique potentials is $E+k$. The converse holds the same way. Since M is sufficiently large, the edge (u_1,u_2) in CI will always have both ends labeled differently. So given a labeling in CI, we can define a cut in G by the subset of vertices v for which v_1 is labeled 0 in CI.

■

At this point we note that our collective inference objective is related to various bodies of work on joint inference, primarily in computer vision tasks. For the sake of continuity we defer our discussion of all related work to Section 8 and instead present our framework for maximizing the collective inference objective.

3. Cluster Graph based Collective Inference Framework

Having established that optimizing the objective in Equation 2 is NP-hard, one natural choice for approximating it is ordinary pairwise belief propagation on the collective graphical model. This involves passing messages along edges inside the chains, as well as along the clique edges. However this approach is not suitable due to many reasons. First, some symmetric potentials like the Majority potential cannot be decomposed along the clique edges, so we cannot compute the corresponding messages. Second, the approach does not exploit the special nature of the clusters and the symmetric potentials. Third, this approach is not tenable if we wish to extend the framework to capture associativities of more general properties of the kind discussed in Section 6.

Hence we adopt message passing on a special cluster graph where every chain and clique corresponds to a cluster. The clique cluster for a unigram is adjacent to all the chains that contain that unigram, as shown via singleton separator vertices in Figure 1(e). This setup of message passing on the cluster graph allows us to exploit potential-specific algorithms at the cliques, and at the same time work with any arbitrary symmetric clique potentials. Cluster graphs have been used effectively for message passing elsewhere as well (Yedidia et al., 2003; Duchi et al., 2007).

Let $m_{k\to t}$ and $m_{t\to k}$ denote message vectors from chain k to an incident clique t and vice-versa. A chain is said to be incident on a clique if there exists a position j in chain k which matches term t, that is, $(k, j) \in D(t)$. We assume that $D(t)$ is created such that from any chain only one position belongs to it. We next discuss how these messages are computed.

3.1 Message from an Instance to a Clique

The message $m_{k\to t}(y)$ for a $(k, j) \in D(t)$ is given by:

$$m_{k\to t}(y) = \max_{\mathbf{y}^k : y^k_j = y} \left(\sum_{i=1}^{|\mathbf{x}^k|} \phi^k(y^k_i, y^k_{i-1}) + \sum_{t' \neq t \in \mathcal{T},\, (k,j') \in D(t')} m_{t'\to k}(y^k_{j'}) \right).$$

To compute $m_{k\to t}(y)$, we need to absorb the incoming messages from other incident cliques $t' \neq t$, and do the maximization while freezing the label of position j in chain k to y for a $(k, j) \in D(t)$. A clique t' is incident to chain k via only a singleton vertex so we can easily absorb the message $m_{t'\to k}$ by including it in the node potential of this vertex. After absorption, $m_{k\to t}$ can be computed using the same inference algorithm applicable to the chain.

3.2 Message from a Clique to an Instance

The more interesting message computation is that for $m_{t\to k}$ given a $(k, j) \in D(t)$. Let the clique t have n vertices. Then the message computation can be written as:

$$m_{t\to k}(y) = \max_{(y_1,\ldots,y_k=y,\ldots,y_n)} \sum_{(k',j')\in D(t)\ k'\neq k} m_{k'\to t}(y_{k'}) + \mathrm{C}_t(\{y_1,\ldots,y_n\}). \tag{3}$$

The maximization in Equation 3 can be re-written as

$$-m_{k\to t}(y) + \max_{(y_1,\ldots y_k=y,\ldots,y_n)} \left(\sum_{(k',j')\in D(t)} m_{k'\to t}(y_{k'}) + \mathrm{C}_t(\{y_1,\ldots,y_n\}) \right). \tag{4}$$

The maximization term is an instance of the general *clique inference problem* defined as:

Definition 3 (Clique Inference) *Given a clique over n vertices, with a symmetric clique potential $C(y_1,\ldots,y_n)$, and vertex potentials ψ_{jy} for all $j \leq n$ and labels y. Compute a labeling of the clique vertices that maximizes:*

$$\max_{y_1,\ldots,y_n} \sum_{j=1}^{n} \psi_{jy_j} + C(y_1,\ldots,y_n). \tag{5}$$

Thus the second term in Equation 4 can be seen as clique inference by defining $\psi_{jy} \triangleq m_{j\to t}(y)$ and $\mathrm{C} \triangleq \mathrm{C}_t$. To compute $m_{t\to k}(y)$, we can solve the clique inference problem with the easily enforceable constraint $y_k = y$. From now on, we refer to outbound message computation at the cliques as clique inference.

4. Symmetric Clique Potentials

Having established cluster message passing as our collective inference paradigm, and clique inference as our message computation tool, we turn our attention towards various families of symmetric clique potentials. As seen in Section 2, these associative clique potentials depend only on the histogram of label counts $\{n_y|y = 1,\ldots,m\}$ over the clique, n_y being the number of clique vertices labeled y. Thus for ease of notation, we will denote the arguments of a symmetric potential C_t by either the vertex labels $\mathbf{y}_t = (y_1,\ldots,y_n)$ or by its corresponding count histogram $\mathbf{n}(\mathbf{y}_t) = (n_1,\ldots,n_m)$. Here we clarify that $\mathbf{n}$ is used to denote the entire count histogram, n_y to denote the count for label y, while n denotes the clique size. An associative symmetric clique potential is thus maximized when $n_y = n$ for some y, that is, one label is given to all the clique vertices.

We consider specific families of clique potentials, many of which are currently used in real-life tasks. In Section 5 we will look at various potential-specific exact and approximate clique inference algorithms that exploit the specific structure of the potential at a clique.

In particular, we consider the three types of symmetric clique potentials listed in Table 1. They differ in the manner in which they reward skew in the count histogram $\mathbf{n}$.

4.1 MAX Clique Potentials

These clique potentials are of the form:

$$\mathrm{C}(n_1,\ldots,n_m) = \max_y f_y(n_y).$$

Name	Form	Remarks
MAX	$\max_y f_y(n_y)$	f_y is a non-decreasing function.
SUM	$\sum_y f_y(n_y)$	f_y non-decreasing. Includes Potts = $\lambda \sum_y n_y^2$.
MAJORITY	$f_a(\mathbf{n})$, where $a = \text{argmax}_y n_y$	A popular form is $f_a(\mathbf{n}) = \sum_y w_{ay} n_y$.

Table 1: Three kinds of symmetric clique potentials considered in this paper. $\mathbf{n} = (n_1, \ldots, n_m)$ denotes the counts of various labels among the clique vertices.

for arbitrary non-decreasing functions f_y. When $f_y(n_y) \triangleq n_y$, we get the *makespan* clique potential which has roots in the job-scheduling literature. This potential depends only on the biggest label count and ignores the other labels. Truncated version of this potential have been recently used in computer vision (Kohli et al., 2009).

In Section 5.1, we present the α-pass algorithm that computes messages for MAX clique potentials exactly in $O(mn \log n)$ time. MAX potentials are tractable and relatively simpler potentials, but most importantly, they provide key insights to deal with the more complex SUM potentials.

4.2 SUM Clique Potentials

SUM clique potentials are of the form:

$$\text{C}(n_1, \ldots, n_m) = \sum_y f_y(n_y).$$

This family of potentials includes functions that aggregate the histogram skew over bins, for example, the entropy potential where $f_y(n_y) \propto n_y \log n_y$. One very interesting member is the case when the well-known Potts model is applied homogeneously on all edges of a clique. Let $\lambda > 0$ be the pairwise reward of assigning the same label to two nodes of an edge. The summation of these terms over a clique is equivalent (up to a constant) to the clique potential:

$$\text{C}^{\text{Potts}}(n_1, \ldots, n_m) = \lambda \sum_y n_y^2.$$

This corresponds to the gini entropy of the histogram. We will show that the clique inference problem is NP-hard with the above potential and provide a $\frac{13}{15}$-approximation in Section 5.

We note that the traditional usage of Potts is in a minimization setting, that is, edges are penalized by some cost γ if their end vertices are labeled differently. The corresponding cost version of Potts is then $\text{C}^{\min}(y_1, \ldots, y_n; \gamma) \triangleq \gamma \sum_{j>i} \delta(y_i \neq y_j)$. It is easy to see that these two versions are related by:

$$\text{C}^{\text{Potts}}(y_1, \ldots, y_n; \lambda) = \lambda n^2 - \text{C}^{\min}(y_1, \ldots, y_n; 2\lambda).$$

We show that our algorithm provides a $\frac{3}{2}$ approximation for the minimization version of the problem.

4.3 MAJORITY Clique Potentials

A MAJORITY potential is defined as:

$$\text{C}(n_1, \ldots, n_m) = f_a(\mathbf{n}), \; a = \text{argmax}_y \, n_y. \tag{6}$$

An important special subclass is the linear majority potentials defined as:

$$\mathrm{C}^{\mathrm{Maj}} = \sum_y w_{ay} n_y, \; a = \mathrm{argmax}_y \; n_y.$$

This potential has been used for a variety of tasks such as link-based classification of web-pages (Lu and Getoor, 2003) and named-entity extraction (Krishnan and Manning, 2006). The role of the parameters w_{ay} is to capture the co-existence of some label pairs in the same clique. Co-existence allows us to downplay 'strict associativity' viz. it permits a few cliques vertices to have similar but not necessarily the same labels. For example, consider a clique made from occurrences of the unigram 'America'. However, some occurrences of America correspond to Location, while others might correspond to an Organization, say Bank of America. Also, it is rare for a location name to be shared with a person name. This can be captured by allowing a higher value of $w_{\alpha y}$ for the (Location, Organization) pair than for the (Location, Person) pair.

Unlike Potts potential, MAJORITY potential cannot be represented using edge potentials. We will present an exact polynomial time algorithm and several efficient approximations in Section 5.3.

5. Algorithms for Clique Inference

We will use $F(y_1,\dots,y_n)$ to denote the clique inference objective in Equation 5. As short-hand, we will denote $F(y_1,\dots,y_n)$ by $F(\mathbf{y}) = \psi(\mathbf{y}) + \mathrm{C}(\mathbf{y})$, where $\psi(\mathbf{y})$ is the vertex score (i.e., node potential) of the clique labeling $\mathbf{y}$ and the second term is the clique score (i.e., clique potential). Wlog assume that all the vertex terms ψ_{iy} are positive. Otherwise a constant can be added to all of them and that will not affect the maximization. The MAP clique labeling will be denoted by $\mathbf{y}^*$, and $\hat{\mathbf{y}}$ will denote a possibly sub-optimal labeling.

We show that the clique inference is easy for *any* $\mathrm{C}()$ with just two labels and in Section 5.1 we present an exact algorithm called α-pass for this case. We show that the same algorithm generalizes to give an exact solution for MAX potentials. We address the SUM and MAJORITY clique potentials respectively in Sections 5.2 and 5.3. Finally, in Section 5.4 we show how to extend the clique inference algorithms to efficiently batch the computation of multiple max-marginals.

5.1 α-pass Algorithm

We begin with MAX potentials. Recall that a MAX potential is of the form $\mathrm{C}(\mathbf{n}(\mathbf{y})) = \max_y f_y(n_y)$. We propose an exact inference procedure called α-pass (Algorithm 1) for such potentials.

The α-pass algorithm guesses that the dominant label in $\mathbf{y}^*$ is α, with a count of k. Of course we do not know α or k so all (α,k) combinations are tried out. For each (α,k) combination, α-pass computes the best k vertices to assign the label α. These k vertices are obtained by sorting all the vertices according to the criteria $\psi_{.\alpha} - \max_{\beta\neq\alpha}\psi_{.\beta}$, and picking the top-$k$ vertices. Every remaining vertex u is labeled with the best non-α label as per the vertex score, that is, with $\arg\max_\beta \psi_{u\beta}$. Let $\hat{\mathbf{y}}^{\alpha k}$ denote the clique labeling thus obtained in the $(\alpha,k)^{th}$ combination. Trying out all (α,k) combinations, α-pass returns the $\hat{\mathbf{y}}^{\alpha k}$ whose score $F(\hat{\mathbf{y}}^{\alpha k})$ is the highest.

It is straightforward to see that α-pass runs in $O(mn\log n)$ time by incrementally computing $F(\hat{\mathbf{y}}^{\alpha k})$ from $F(\hat{\mathbf{y}}^{\alpha(k-1)})$, that is, for each α, we can sort the vertices just once for all $k = 1,\dots,n$. For clique potentials that are decomposable over the edges, as in Potts, this runtime is much better than ordinary belief propagation, which would cost $O(m^2n^2)$.

We now look at properties of α-pass.

```
Input: Vertex scores $\psi$, Clique Potential C
Output: Labeling $\hat{\mathbf{y}}$
Best = $-\infty$;
foreach label $\alpha \in \{1,\ldots,m\}$ do
    Sort the vertices in descending order according to the metric $\psi_{.\alpha} - \max_{\beta\neq\alpha}\psi_{.\beta}$;
    foreach $k \in \{1,\ldots,n\}$ do
        Assign the first $k$ sorted vertices the label $\alpha$;
        Assign the remaining vertices their individual best non-$\alpha$ label;
        s $\leftarrow$ score of this labeling under $F()$;
        if $s > Best$ then
            Best $\leftarrow$ s;
            $\hat{\mathbf{y}}$ $\leftarrow$ current labeling;
        end
    end
end
return $\hat{\mathbf{y}}$;
```

Algorithm 1: The α-pass algorithm

Claim 5.1 *Assignment $\hat{\mathbf{y}}^{\alpha k}$ has the maximum vertex score over all $\mathbf{y}$ where k vertices are assigned label α, that is, $\psi(\hat{\mathbf{y}}^{\alpha k}) = max_{\mathbf{y}:n_\alpha(\mathbf{y})=k}\psi(\mathbf{y})$.*

Claim 5.2 *For* MAX *potentials, $C(\hat{\mathbf{y}}^{\alpha k}) \geq f_\alpha(k)$.*

Proof Label α has a count of k and the MAX potential considers the maximum over all label counts, which is at least k. ■

Theorem 4 *The α-pass algorithm solves the clique inference problem exactly for* MAX *clique potentials in $O(mn\log n)$ time.*

Proof Let $\mathbf{y}^*$ be the true MAP and let $\beta = \text{argmax}_y f_y(n_y(\mathbf{y}^*))$, $\ell = n_\beta(\mathbf{y}^*)$. Let $\hat{\mathbf{y}}$ be the labeling returned by α-pass. We have:

$$\begin{aligned}
F(\hat{\mathbf{y}}) &= \max_{1\leq\alpha\leq m, 1\leq k\leq n} F(\hat{\mathbf{y}}^{\alpha k}) \\
&\geq F(\hat{\mathbf{y}}^{\beta\ell}) \\
&= \psi(\hat{\mathbf{y}}^{\beta\ell}) + \mathrm{C}(\hat{\mathbf{y}}^{\beta\ell}) \\
&\geq \psi(\hat{\mathbf{y}}^{\beta\ell}) + f_\beta(\ell) \quad \text{(by Claim 5.2)} \\
&= \psi(\hat{\mathbf{y}}^{\beta\ell}) + \mathrm{C}(\mathbf{y}^*) \\
&\geq \psi(\mathbf{y}^*) + \mathrm{C}(\mathbf{y}^*) \quad \text{(by Claim 5.1)} \\
&= F(\mathbf{y}^*).
\end{aligned}$$

■

Although we had initially designed the α-pass algorithm for MAX potentials, a similar argument can be used to show that α-pass performs exact clique inference for *any* arbitrary symmetric potential when we have two labels.

Claim 5.3 *α-pass is exact for arbitrary symmetric potentials over two labels.*

Proof Let the MAP $\mathbf{y}^*$ have label counts n_1 and $n-n_1$, and a vertex score of $\psi(\mathbf{y}^*)$. Let $\hat{\mathbf{y}}$ be the labeling returned by α-pass. Then we get $F(\hat{\mathbf{y}}) \geq F(\hat{\mathbf{y}}^{1n_1}) = \psi(\hat{\mathbf{y}}^{1n_1}) + C(n_1, n-n_1) \geq \psi(\mathbf{y}^*) + C(n_1, n-n_1) = F(\mathbf{y}^*)$. Since C is arbitrary, the result follows. ■

We also note that in some real-life tasks the vertex terms tend to heavily dominate the clique term. In such a case too, α-pass will provide an exact solution. This follows immediately from Claim 5.1 and the observation that $F(\mathbf{y}) \approx \psi(\mathbf{y})$.

5.2 Clique Inference for SUM Potentials

We focus on the Potts potential, the most popular member of the SUM family. Potts potential is given by $\mathrm{C}^{\text{Potts}}(\mathbf{y}) = \lambda \sum_y n_y^2$ and the clique inference objective is:

$$\max_{y_1,\ldots,y_n} \sum_{j=1}^{n} \psi_{jy_j} + \lambda \sum_y n_y^2. \tag{7}$$

In the above, $\lambda > 0$ since we are interested in the associative case which encourages agreement among labels. It is well known that inference with Potts potentials over general graphs, is NP-hard (Boykov et al., 2001) when number of labels is > 2 even with $\lambda > 0$. So we will first show that the task remains NP-hard even for cliques. Then, we will prove that α-pass achieves an approximation ratio of $\frac{4}{5}$, followed by a more complex proof for a better and tight approximation ratio of $\frac{13}{15}$. We will then generalize the α-pass algorithm, which will result in an improved ratio of $\frac{8}{9}$ at a cost of higher runtime. In contrast, the α-expansion algorithm of Boykov et al. (2001) will be shown to have a ratio of only $\frac{1}{2}$.

Theorem 5 *When $C(\mathbf{y}) = \lambda \sum_y n_y^2, \lambda > 0$, clique inference is NP-hard.*

Proof We prove hardness by reduction from the NP-complete *Exact Cover by 3-sets* problem (Papadimitriou and Steiglitz, 1982). In an instance of Exact Cover by 3-sets, we are given a universe U of elements, a set S of subsets of U where each subset has three elements, and the goal is to find $S' \subseteq S$ that covers U, while minimizing $|S'|$. We create an instance of clique inference as follows. We let elements of U correspond to vertices, and each set in S to a label. Assign $\psi_{iy} = 2n\lambda$ if element i belongs to set y, and zero otherwise (set $\lambda > 0$ arbitrarily). Consider the problem of deciding if exactly $\frac{n}{3}$ out of m subsets cover U. The MAP score in the constructed clique inference instance will be $(2n^2 + 3^2\frac{n}{3})\lambda$ iff we can find an exact cover. ■

The above proof establishes that there cannot be an algorithm that is polynomial in both n and m. But we have not ruled out algorithms with complexity that is polynomial in n but exponential in m, say of the form $O(2^m n^k)$ for a constant k.

We next analyze the approximation guarantees provided by the α-pass algorithm. We first present an easy proof for a weaker bound of $\frac{4}{5}$ and then move on to a more detailed proof for $\frac{13}{15}$. Recall that the optimal labeling is $\mathbf{y}^*$ and the labeling output by α-pass is $\hat{\mathbf{y}}$.

Theorem 6 $F(\hat{\mathbf{y}}) \geq \frac{4}{5}F(\mathbf{y}^*)$.

Proof Without loss of generality assume that the label counts of $\mathbf{y}^*$ are $n_1 \geq n_2 \geq \ldots \geq n_m$. Then $\mathrm{C}(\mathbf{y}^*) = \lambda \sum_y n_y^2 \leq \lambda n_1 \sum_y n_y = \lambda n n_1$.

$$\begin{aligned}
F(\hat{\mathbf{y}}) &\geq F(\hat{\mathbf{y}}^{1n_1}) = \psi(\hat{\mathbf{y}}^{1n_1}) + \mathrm{C}(\hat{\mathbf{y}}^{1n_1}) \\
&\geq \psi(\mathbf{y}^*) + \mathrm{C}(\hat{\mathbf{y}}^{1n_1}) \quad \text{(from Claim 5.1)} \\
&\geq \psi(\mathbf{y}^*) + \lambda n_1^2 \quad \text{(since } \lambda > 0\text{)} \\
&\geq \psi(\mathbf{y}^*) + \mathrm{C}(\mathbf{y}^*) - \lambda n_1 n + \lambda n_1^2 \\
&\geq F(\mathbf{y}^*) - \lambda n^2/4. \qquad (8)
\end{aligned}$$

Now consider the two cases where $F(\mathbf{y}^*) \geq \frac{5}{4}\lambda n^2$ and $F(\mathbf{y}^*) < \frac{5}{4}\lambda n^2$. For the first case we get from above that $F(\hat{\mathbf{y}}) \geq F(\mathbf{y}^*) - \lambda n^2/4 \geq \frac{4}{5}F(\mathbf{y}^*)$. For the second case, we know that the score $F(\hat{\mathbf{y}}^{mn})$ where we assign all vertices the last label is at least λn^2 (because all vertex scores are assumed to be non-negative) and thus $F(\hat{\mathbf{y}}) \geq \frac{4}{5}F(\mathbf{y}^*)$. ■

We stress that non-negativity of the vertex scores is important for the multiplicative bound of $\frac{4}{5}$, else one can construct examples where $F(\mathbf{y}^*) = 0$. At the same time, Equation 8 provides a useful additive bound that holds for arbitrary vertex scores.

We now state the more involved proof for showing that α-pass actually provides a tight approximation bound of $\frac{13}{15}$ for clique inference with Potts when the vertex scores are non-negative.

Theorem 7 $F(\hat{\mathbf{y}}) \geq \frac{13}{15}F(\mathbf{y}^*)$. *Further, this ratio is tight.*

Proof See Appendix A. ■

We next present a generalization of the α-pass algorithm that provides provably better guarantees for Potts.

5.2.1 GENERALIZED α-PASS ALGORITHM

In α-pass, for each label α, we go over each count k and find the best vertex score with k vertices assigned label α. We generalize this to go over all label subsets of size no more than q, a parameter of the algorithm that is fixed based on the desired approximation guarantee.

For each label subset $A \subseteq \{1, \ldots, m\}$ of size no more than q, and for each count k, maximize vertex scores with exactly k vertices assigned a label from A. For this, we sort the vertices in decreasing order of $\max_{\alpha \in A} \psi_{i\alpha} - \max_{y \notin A} \psi_{iy}$, assign the top k vertices their best label in A and the remaining their best label not in A. The best solution over all combinations (A, k) with $|A| \leq q$ is returned as the final labeling $\hat{\mathbf{y}}$. It is easy to see that this algorithm reduces to α-pass when $q = 1$.

The complexity of this algorithm is $O(nm^q \log n)$ because there are $\binom{m}{q}$ choices for the set A. In practice, we can use heuristics to prune the number of label subsets. Further, we can make the following claims about the quality of its output.

Theorem 8 *Generalized α-pass enjoys an approximation bound of $\frac{8}{9}$, that is, $F(\hat{\mathbf{y}}) \geq \frac{8}{9}F(\mathbf{y}^*)$.*

Proof The bound is achieved with $q = 2$. We provide the details in Appendix A. ■

We *conjecture* that the bound for general q is $\frac{4q}{4q+1}$. This bound is not tight as for $q = 1$ we have already shown that the $\frac{4}{5}$ bound can be tightened to $\frac{13}{15}$. With $q = 2$ we get a bound of $\frac{8}{9}$ which is better than $\frac{13}{15}$.

5.2.2 α-EXPANSION ALGORITHM

In general graphs, a popular method that provides the approximation guarantee of 1/2 for the Potts model is the graph-cut based α-expansion algorithm (Boykov et al., 2001). We explore the behavior of this algorithm for clique inference with Potts potentials.

In this scheme, we start with any initial labeling—for example, all vertices are assigned the first label, as suggested in Boykov et al. (2001). Next, for each label α we perform an α-expansion phase where we switch the labeling of an optimal set of vertices from their current label to α. We repeat this in a round over the m labels, until no vertices switch their labeling in a complete round.

For graphs whose edge potentials form a metric, an optimal α-expansion move is based on the use of the mincut algorithm of Boykov et al. (2001) which for the case of cliques can be $O(n^3)$. We next show how to perform optimal α-expansion moves more efficiently (in $O(mn^2)$ time) for all kinds of SUM potentials.
An α-expansion move: Let $\tilde{\mathbf{y}}$ be the labeling at the start of this move. For each label $y \neq \alpha$ create a sorted list S_y of vertices assigned y in $\tilde{\mathbf{y}}$ in decreasing order of $\psi_{i\alpha} - \psi_{iy}$. If in an optimal move, we move k_y vertices from y to α, then it is clear that we need to pick the top k_y vertices from S_y. Let r_i be the rank of a vertex i in S_y. Our remaining task is to decide the optimal number k_y to take from each S_y. We find these using dynamic programming. Without loss of generality assume $\alpha = m$. Let $D_j(k)$ denote the best score when k vertices with current labels in $1 \dots j$ switch to α. We compute

$$D_j(k) = \max_{l \le k,\ l \le n_j(\tilde{\mathbf{y}})} D_{j-1}(k-l) + f_j(n_j(\tilde{\mathbf{y}}) - l) + \sum_{i':r_{i'} \le l} \psi_{i'\alpha} + \sum_{i':r_{i'} > l} \psi_{i'j},$$

where $n_j()$ is the usual cardinality of label j in the labeling. Now we can find the optimal number of vertices to switch to α as $\text{argmax}_{k \le n - n_\alpha(\tilde{\mathbf{y}})} D_{m-1}(k) + f_\alpha(k + n_\alpha(\tilde{\mathbf{y}}))$.

5.2.3 COMPARISON WITH EXISTING APPROXIMATION BOUNDS

As mentioned earlier, the C^{Potts} clique potential is equivalent to the sum of Potts potentials over edges of a complete graph. For arbitrary graphs with homogeneous Potts potential on edges, the alpha expansion algorithm of Boykov et al. (2001) and the LP relaxation algorithm of Kleinberg and Tardos (2002) provide a factor of 2 approximation guarantee for a minimization version of the objective. For cliques with homogeneous edge potentials, their objective reduces to an energy-based formulation (using their notation):

$$\min_{\mathbf{y}} \sum_j \theta_j(y_j) + \gamma \sum_{i,j>i} \delta(y_j \neq y_i) \equiv \min_{\mathbf{y}} \sum_j \theta_j(y_j) + \frac{\gamma}{2}(n^2 - \sum_y n_y^2), \tag{9}$$

where $\theta_j(y_j)$ denotes node energy and is assumed to be positive and γ denotes the homogeneous Potts parameter.

First we show that α-expansion provides an approximation bound of 2 even for the special case of cliques. Symmetrically, we also show that the α-expansion algorithm provides a bound of $\frac{1}{2}$ for our original max-version of the clique inference problem. Next, we show that α-pass provides a bound of $\frac{3}{2}$ for even the minimization objective above. Thus, for both the minimization and maximization objectives, α-pass improves upon the guarantees obtained by existing algorithms.

Theorem 9 *The α-expansion algorithm provides no better approximation guarantee than 2 for the minimization objective (9).*

Proof An instance where this occurs is as follows. There are n nodes and $n+1$ labels $y_0,\ldots,y_n$. The vertex energy for (x_i,y_0) is $2n-2$ and for (x_i,y_i) it is 0. All other vertex energies are ∞, and $\gamma=1$. Suppose initially all nodes are assigned label y_0. Then the clique energy is 0 and vertex energy is $n(2n-2)$. Now considering any other label $y_i, i>=1$, the optimal set of nodes to switch is just x_i. But this reduces vertex energy by $2n-2$ but increases clique energy by the same amount, so no switching is done. However the optimal solution has x_i labeled with y_i, which has a total energy of n^2-n. ■

Theorem 10 *The α-expansion algorithm provides no better approximation guarantee than 1/2 for the maximization objective (7).*

Proof Consider an instance where $m=\sqrt{n}+1$, and $\lambda=1$. Let $\psi_{u1}=2\sqrt{n}$ for all u. Divide the vertices into $\sqrt{n}$ groups of size $\sqrt{n}$ each, and let $\psi_{u,i+1}=2n$ for every vertex u in the i^{th} group. All other vertex scores are zero. Consider the solution where every vertex is assigned label 1. This labeling is locally optimal wrt any α-expansion move, and its score is $n^2(1+2/\sqrt{n})$. However, the exact solution assigns every vertex group its label, with a score $n^2(2+1/\sqrt{n})$, thus giving a ratio of $1/2$ in the limit. ■

Theorem 11 *The α-pass algorithm achieves an approximation ratio of $\frac{3}{2}$ for the minimization objective 9.*

Proof Suppose $k=\max_y n_y$ is the highest count in the optimal labeling. Consider two cases, $k\geq n/2$ and $k<n/2$.

If $k\geq n/2$, the clique energy is at least $\gamma(n^2-(k^2+(n-k)^2))=\gamma 2k(n-k)$. The clique energy in α-pass is at most $\gamma(2n^2-k^2-(n-k))=\gamma(n-k)(n+k-1)$. The vertex energy in the optimal labeling cannot be smaller than that in α-pass. Since $k>=n/2, (n+k-1)/2k\leq 3/2$.

If $k\leq n/2$, then the clique energy in the optimal labeling is at least $\gamma(n^2-nk)=\gamma n(n-k)$. Therefore the ratio is again at most $(n+k-1)/n\leq 3/2$. ■

As expected, generalized α-pass provides a superior approximation ratio for Objective 9.

Theorem 12 *The generalized α-pass algorithm with two labels ($q=2$) achieves an approximation ratio of $\frac{1+\sqrt{2}}{2}$ for objective 9.*

Proof We omit the complete proof as it is quite detailed and give only a brief sketch. Wlog assume that 1 and 2 are the two most dominant labels in the optimal labeling OPT, with counts $n_1, n_2\leq n_1$. Consider two solutions—one given by α-pass (i.e., $q=1$) when label 1 has count n_1, and another given by generalized α-pass with $q=2$ when counts of labels 1 and 2 are n_1+n_2. The node energies of these two labelings cannot be worse than that of OPT (Claim 5.1). So the approximation bound depends only on the ratios of the clique energies. Therefore wlog assume that $\gamma=2$ in Objective 9. The clique energies of the two solutions cannot exceed $(n^2-n_1^2)$ and $(n^2-\frac{(n_1+n_2)^2}{2})$ respectively. We can now work out the two cases whether $n_1^2\leq(n_1+n_2)^2/2$ or not, and get the desired approximation bound through contradictions. The analysis also shows that the bound is tight, and is achieved when

$n_1 = n/\sqrt{2}$ and $n_2 = n - n/\sqrt{2}$. ■

5.2.4 ENTROPY POTENTIALS AND THE α-PASS ALGORITHM

As an aside, let us explore the behavior of α-pass on another family of additive potentials—entropy potentials. Entropy potentials are of the form:

$$\mathrm{C}(\mathbf{n}(\mathbf{y})) = \lambda \sum_y n_y \log n_y, \text{ where } \lambda > 0.$$

The main reason α-pass provides a good bound for Potts potentials is that it guarantees a clique potential of at least n_1^2 where n_1 is the count of the most dominant label in the optimal solution $\mathbf{y}^*$. The quadratic term compensates for possible sub-optimality of counts of other labels. If we had a sub-quadratic term instead, say $n_1 \log n_1$ for the entropy potentials, the same bound would not have held. In fact the following theorem shows that for entropy potentials, even though α-pass guarantees a clique potential of at least $n_1 \log n_1$, that is not enough to provide a good approximation ratio.

Theorem 13 *α-pass does not provide a bound better than $\frac{1}{2}$ for entropy potentials.*

Proof Consider a counter example where there are $m = n + \log n$ labels. Divide the labels into two sets—A with $\log n$ labels and B with n labels. The vertex scores are as follows: the vertices are divided into $\log n$ chunks of size $n/\log n$ each. If the j^{th} vertex lies in the y^{th} chunk, then let it have a vertex score of $\log n$ with label y in A and a vertex score of $\log n + \varepsilon$ with the j^{th} label in B. Let all other vertex scores be zero. Also, let $\lambda = 1$.

Consider the labeling which assigns the y^{th} label in A to the y^{th} chunk. Its score is $2n \log n - n \log\log n$. Now consider α-pass, with $\alpha \in A$. Initially vertex y will be set to the y^{th} label in B. The best labeling found by α-pass will assign every vertex to α, for a total score of roughly $n + n \log n$. If $\alpha \in B$, then again the best labeling will assign everything to α for a total score of roughly $(n+1)\log n$.

Thus the bound is no better than $\frac{1}{2}$ as $n \to \infty$. ■

Thus, α-pass provides good approximations when the clique potential is heavily dominated by the most dominating label. We now look at MAJORITY potentials, which are linear in the counts $\{n_y\}_y$. Looking at Theorem 13, we expect that α-pass will not have decent approximation guarantees for MAJORITY. This is indeed the case. We will prove in Section 5.3 that neither α-pass nor a natural modification of α-pass enjoy good approximation guarantees.

5.3 Clique Inference for MAJORITY Potentials

Recall that MAJORITY potentials are of the form $\mathrm{C} = f_a(\mathbf{n}),\ a = \mathrm{argmax}_y\, n_y$. We consider linear majority potentials where $f_a(\mathbf{n}) = \sum_y w_{ay} n_y$. The matrix $W = \{w_{yy'}\}$ is not necessarily diagonally dominant or symmetric.

We show that exact MAP for linear majority potentials can be found in polynomial time. We also present a modification to the α-pass algorithm to serve as an efficient heuristic, but without approximation guarantees. Then we present a Lagrangian relaxation based approximation, whose runtime is competitive with α-pass, but which provides much more accurate solutions.

5.3.1 MODIFIED α-PASS ALGORITHM

Assume that we magically know the majority label α in advance. Then for linear majority potentials, we can incorporate the linear clique term $w_{\alpha y}n_y$ in the various vertex scores, and this leads to the following natural modifications to the α-pass algorithm: (a) While making iterations for the label α, sort the vertices according to the modified metric $\psi_{i\alpha} + w_{\alpha\alpha} - \max_{y \neq \alpha}(\psi_{iy} + w_{\alpha y})$, and (b) While sweeping the list for α with varying values of k, discard all candidate solutions whose majority label is not α.

However even after these modifications, α-pass does not provide the same approximation guarantee as for homogeneous Potts potentials, as we prove next.

Theorem 14 *The modified α-pass algorithm cannot have an approximation ratio better than $\frac{1}{2}$ on linear majority potentials with arbitrary W.*

Proof Consider the degenerate example where all vertex scores are zero. Let β and γ be two fixed labels and let the matrix W be defined as follows: $w_{\beta\gamma} = M + \varepsilon$, $w_{\beta y} = M\ \forall y \neq \beta, \gamma$ and all the other entries in W are zero.

In modified α-pass, when $\alpha \neq \beta$, the labeling returned will have a zero score. When $\alpha = \beta$, all vertices will prefer the label γ, so α-pass will have to assign exactly $n/2$ vertices to β to make it the majority label, thus returning a score of $\frac{(M+\varepsilon)n}{2}$. However, consider the labeling which assigns n/m vertices to each value, with a score of $(m-1)Mn/m$. Hence the approximation ratio cannot be better than $\frac{1}{2}$. ■

5.3.2 EXACT ALGORITHM

Since MAJORITY potentials are linear, we can pose the optimization problem in terms of Integer Programs (IPs). Assume that we know the majority label α. Then, the optimization problem corresponds to the IP:

$$\begin{aligned} \max_{\mathbf{z}} \sum_{i,y} (\psi_{iy} + w_{\alpha y}) z_{iy}, \\ \text{s.t. } \forall y: \ \sum_i z_{iy} \leq \sum_i z_{i\alpha}, \\ \forall i: \ \sum_y z_{iy} = 1,\ z_{iy} \in \{0,1\}. \end{aligned} \tag{10}$$

We can solve m such IPs by guessing various labels as the majority label, and reporting the best overall labeling as the output. However, Equation 10 cannot be tightly relaxed to a linear program. This can be easily shown using a counter example: Consider a 3-vertex, 3-label clique with a zero W matrix. Let the vertex score vectors be $\psi_0 = (1,4,0)$, $\psi_1 = (4,0,4)$, $\psi_2 = (3,4,0)$. While solving for $\alpha = 0$, the best IP labeling is $1,0,0$ with a score of 11. However the LP relaxation has the solution $\mathbf{z} = (0,1,0;1,0,0;1/2,1/2,0)$ with a score of 11.5.

This issue can be resolved by making the constraint matrix totally unimodular as follows. Guess the majority label α, its count $k = n_\alpha$, and solve the following IP:

$$\begin{aligned} &\max_{\mathbf{z}} \sum_{i,y} (\psi_{iy} + w_{\alpha y}) z_{iy}, \\ &\text{s.t. } \forall y \neq \alpha : \sum_i z_{iy} \leq k, \\ &\sum_i z_{i\alpha} = k, \\ \forall i : &\sum_y z_{iy} = 1,\ z_{iy} \in \{0,1\}. \end{aligned} \tag{11}$$

This IP solves the degree constrained bipartite matching problem, which can be solved exactly in polynomial time. Indeed, it can be shown that the constraint matrix of this IP is totally unimodular, so its LP relaxation will have an integral solution. We refer the reader to Gupta et al. (2009) for the details. Thus we solve $O(mn)$ such problems by varying α and k, and report the best solution. We believe that the above LP is concave in k, so the system for a fixed α should be efficiently solvable using golden section search.

5.3.3 LAGRANGIAN RELAXATION BASED ALGORITHM FOR MAJORITY POTENTIALS

Solving the linear system in Equation 11 is very expensive because we need to solve $O(mn)$ LPs, whereas the system in Equation 10 cannot be solved exactly using a linear relaxation. Here, we look at a Lagrangian Relaxation based approach (LR), where we solve the system in Equation 10 but bypass the troublesome constraint $\forall y \neq \alpha : \sum_i z_{iy} \leq \sum_i z_{i\alpha}$.

We use Lagrangian relaxation to move this constraint to the objective function. Any violation of this constraint is penalized by a positive penalty term. Consider the following modified program, also called the Lagrangian:

$$\begin{aligned} L(\gamma) = L(\gamma_1, \ldots, \gamma_m) = \max_{\mathbf{z}} \sum_{i,y} (\psi_{iy} + w_{\alpha y}) z_{iy} \quad &+ \quad \sum_y \gamma_y \Big(\sum_i z_{i\alpha} - \sum_i z_{iy}\Big), \\ \text{s.t. } \forall i : \sum_y z_{iy} = 1, \qquad &z_{iy} \in \{0,1\}. \end{aligned} \tag{12}$$

For $\gamma \geq \mathbf{0}$, and feasible $\mathbf{z}$, $L(\gamma)$ is an upper bound for our objective in Equation 10. Thus, we compute the lowest such upper bound:

$$L^* = \min_{\gamma \geq \mathbf{0}} L(\gamma). \tag{13}$$

Further, the penalty term in Equation 12 is linear in $\mathbf{z}$, so we can merge it with the first term to get another set of modified vertex potentials:

$$\psi^\alpha_{iy} \triangleq \psi_{iy} + w_{\alpha y} - \gamma_y + \left\{ \begin{array}{ll} \sum_{y'} \gamma_{y'} & y = \alpha \\ 0 & y \neq \alpha \end{array} \right. .$$

Equation 12 can now be rewritten in terms of ψ^α, with the only constraint that $\mathbf{z}$ correspond to a valid labeling:

$$\begin{aligned} &\max_{\mathbf{y}} \sum_{i,y} \psi^\alpha_{iy} z_{iy}, \\ \text{s.t. } \forall i : &\sum_y z_{iy} = 1,\ z_{iy} \in \{0,1\}. \end{aligned}$$

Hence, $L(\gamma)$ can be computed by independently assigning each vertex i to its best label viz. $\text{argmax}_y \psi^{\alpha}_{iy}$.

We now focus on computing L^*. We use an iterative approach, beginning with $\gamma = \mathbf{0}$, and carefully choose a new γ at each step to get a non-increasing sequence of $L(\gamma)$'s. We describe the method of choosing a new γ later in this section, and instead outline sufficient conditions for termination and detection of optimality.

Theorem 15 *$\mathbf{z}^*$ and γ^* are optimum solutions to Equations 10 and 13 respectively if they satisfy the conditions:*

$$\forall y: \qquad \sum_i z^*_{iy} \leq \sum_i z^*_{i\alpha}, \tag{14}$$

$$\forall y: \qquad |\gamma^*_y(\sum_i z^*_{iy} - \sum_i z^*_{i\alpha})| = 0. \tag{15}$$

Theorem 15 holds only for fractional $\mathbf{z}^*$. To see how, consider an example with three vertices and two labels. Let $\psi_{i1} + w_{\alpha 1} > \psi_{i2} + w_{\alpha 2}$ for all i and α. During Lagrangian relaxation with $\alpha = 2$, initially $\gamma = \mathbf{1}$ will cause all vertices to be assigned label 1, violating Equation 14. Since the count difference $\sum_i z_{i1} - \sum_i z_{i2} \in \{\pm 1, \pm 2, \pm 3\}$, any non-zero γ_1 will violate Equation 15. Subsequent reduction of γ_1 to zero will again cause the original violation of Equation 14. Consequently, one of Equations 14 and 15 will never be satisfied and the algorithm will oscillate.

To tackle this, we relax Equation 15 to $|\gamma_y(\sum_i z^*_{iy} - \sum_i z^*_{i\alpha})| \leq \varepsilon$, where ε is a small fraction of an upper bound on γ_y. This helps in reporting labelings that respect the majority constraint in Equation 14 and are close to the optimal.

The outline of the algorithm is described in Figure 2.

We now discuss a somewhat conservative approach to select a new γ at every step. We initially attempted subgradient optimization and golden search to compute step direction and sizes for changing γ. However, we ran into various practical difficulties. Subgradient optimization required very careful tweaking of the step size across iterations, an issue exacerbated by the discrete nature of our problem. On the other hand, golden search was too aggressive in practice, leading to many avoidable label flips and consequently many more iterations. So instead we implemented a conservative approach which we describe next.

5.3.4 CONSERVATIVE COORDINATE DESCENT

We perform conservative coordinate descent which avoids large changes and thus too many label flips. Let y be the worst violating label in the current iteration. We will first consider the case when its count exceeds that of α, so that Equation 14 does not hold.

To decrease the count of y, we need to increase γ_y. Let i be a vertex currently assigned y and let $\beta(i)$ be its second most preferred label under the vertex potentials ψ^{α}_i. The vertex $j = \text{argmax}_{i:z_{iy}=1} \psi^{\alpha}_{i\beta(i)} - \psi^{\alpha}_{iy}$ is the easiest to flip. So we increase γ_y just enough to make this flip happen. The new value of γ_y is therefore given by:

$$\gamma_y = \min_{i:z_{iy}=1} \begin{cases} \Delta\psi(i,y,\beta(i)) + \gamma_{\beta(i)} & \beta(i) \neq \alpha \\ \frac{1}{2}(\Delta\psi(i,y,\alpha) - \sum_{y' \neq y} \gamma_{y'}) & \beta(i) = \alpha \end{cases}, \tag{16}$$

where $\Delta\psi(i,y,y')$ denotes $\psi_{iy} + w_{\alpha y} - \psi_{iy'} - w_{\alpha y'}$. It is possible that by flipping vertex j, $\beta(j)$ now violates Equation 14. Moreover, increasing γ_y also increases $\psi^{\alpha}_{i\alpha}$, so some other vertices that are not assigned y may also move to α. However since the change is conservative, we expect this behavior to be limited. In our experiments, we found that this conservative scheme converges much faster than golden search over a variety of data.

We now look at the case when Equation 14 is satisfied by all labels but Equation 15 is violated by some label y. In this scenario, we need to decrease γ_y to decrease the magnitude of the violation. Here too, we conservatively decrease γ_y barely enough to flip one vertex to y. If i is any vertex not assigned label y and $\beta(i)$ is its current label, then the new value of γ_y is given by:

$$\gamma_y = \max_{i:z_{iy}\neq 1} \begin{cases} \Delta\psi(i,y,\beta(i)) + \gamma_{\beta(i)} & \beta(i) \neq \alpha \\ \frac{1}{2}(\Delta\psi(i,y,\alpha) - \sum_{y'\neq y}\gamma_{y'}) & \beta(i) = \alpha \end{cases}. \tag{17}$$

Note that the arguments of Equations 16 and 17 are the same. In this case too, in spite of a conservative move, more than one vertex marked α may flip to some other value, although at most one of them will be flipped to y. As before, the small magnitude of the change restricts this behavior in practice.

```
Input: $\psi, W, \alpha$,maxIters,tolerance
Output: approximately best assignment $\hat{\mathbf{y}}$
$\gamma \leftarrow \mathbf{0}$;
iter $\leftarrow$ 0;
$\hat{\mathbf{z}} \leftarrow$ Assignment with all vertices assigned $\alpha$;
while $iter < maxIters$ do
    Compute $L(\gamma)$ (Equation 12), let $\mathbf{z}$ be the solution;
    if $F(\mathbf{z}) > F(\hat{\mathbf{z}})$ then
        $\hat{\mathbf{z}} \leftarrow \mathbf{z}$;
    end
    $(y, \Delta) \leftarrow$ Worst violator and violation (Equations 14 and 15);
    if $\Delta < tolerance$ then
        We are done, $L^* = L(\gamma)$;
        break;
    else
        Modify $\gamma_y$ using conservative coordinate descent;
    end
    iter $\leftarrow$ iter+1;
end
Construct assignment $\hat{\mathbf{y}}$ from $\hat{\mathbf{z}}$;
return $\hat{\mathbf{y}}$
```

Algorithm 2: LR Algorithm for Majority potentials

5.4 Computing All Node Max-marginals

In this section, we discuss an important optimization that speeds up message-passing in a cluster graph. We show that for symmetric potentials we can compute max-marginals in $O(m^2)$ calls to the clique inference algorithm in practice, as opposed to the expected nm invocations. Cliques can be arbitrarily big, so removing the dependence on n is very helpful in practice.

Our basic strategy is as follows. For label pairs α, β, define a modified clique potential function $C_{\alpha\beta}$ that adds "1" to the count for label β and subtracts "1" from the count of label α:

$$C_{\alpha\beta}(\mathbf{y}) = C(n_1(\mathbf{y}), \ldots, n_\alpha(\mathbf{y}) - 1, \ldots, n_\beta(\mathbf{y}) + 1, \ldots n_m).$$

Find MAP labeling $\mathbf{y}^{\alpha\beta}$ using the modified scoring function $F_{\alpha\beta}(\mathbf{y}) = \psi(\mathbf{y}) + \mathrm{C}_{\alpha\beta}(\mathbf{y})$. Let v be a vertex whose label in $\mathbf{y}^{\alpha\beta}$ is α, then the max marginal $M_{v\beta} \triangleq \max_{\mathbf{y}:y_v=\beta} F(\mathbf{y}) = F_{\alpha\beta}(\mathbf{y}^{\alpha\beta}) - \psi_{v\alpha} + \psi_{v\beta}$. The outgoing message $m_{t\to v}(\beta)$ is then simply $M_{v\beta} - m_{v\to t}(\beta)$ as per Equation 4. A proof of correctness of this optimization is as follows.

Theorem 16 $\max_{\mathbf{y}:y_v=\beta} F(\mathbf{y}) = F_{\alpha\beta}(\mathbf{y}^{\alpha\beta}) - \psi_{v\alpha} + \psi_{v\beta}$.

Proof

$$\begin{aligned}
\max_{\mathbf{y}} F_{\alpha\beta}(\mathbf{y}) &= \max_{\mathbf{y}:y_v=\alpha} F_{\alpha\beta}(\mathbf{y}) \text{ since } v \text{ is labeled } \alpha \text{ in } \mathbf{y}^{\alpha\beta} \\
&= \max_{\mathbf{y}:y_v=\alpha} \psi(\mathbf{y}) + \mathrm{C}_{\alpha\beta}(\mathbf{y}) \\
&= \max_{\mathbf{y}:y_v=\beta} \psi(\mathbf{y}) - \psi_{v\beta} + \psi_{v\alpha} + \mathrm{C}(\mathbf{y}) \\
&= \max_{\mathbf{y}:y_v=\beta} F(\mathbf{y}) - \psi_{v\beta} + \psi_{v\alpha}.
\end{aligned}$$

■

We invoke the above strategy for all label pairs α, β when some vertex in the original MAP gets label α. There is no guarantee that all nm messages would be generated by the above scheme. The ones that are not covered are computed using separate invocations of MAP.

This concludes the description of our various clique inference algorithms and their theoretical properties. The discussion until now had focused on the traditional unigram-clique collective inference model. We now switch tracks and describe an extension of the collective inference framework—one which captures richer kinds of associativity that is often present in the data. We will see that the same cluster message passing setup and collective inference algorithms can be used almost as is in this more general scenario.

6. Properties-based Collective Inference Framework

We broaden the notion of collective inference to encourage richer forms of associativity amongst the labelings of multiple records. This more general framework has applications in domain adaptation. We illustrate this via an example.

Example: Consider extracting bibliographic information from an author's publications homepage using a model trained on a different set of pages. The labels of interest are Title, Author, Venue, and Date. Typically, within each homepage (a domain) we expect consistency in the style of individual publication records. For example, we expect the following properties to be largely uni-valued *inside* a domain:

1. The ordering of labels in the labelings (e.g., Title $\rightarrow$ Author* $\rightarrow$ Venue).
2. The token preceding the Title, or 'Start' if Title occurs at the beginning.
3. The label appearing after the Venue, or 'End' if Venue occurs at the end.
4. Label of the token 'IEEE'.

A Simulator for estimating Railway Line Capacity. *In APORS - 2003*.	(Start, Date, <b>)	Bhardwaj, P. (2001). Delegating Pricing Decisions. *Marketing Science* 20(2). 143-169.	('.', Volume, <li>)
Scheduling Loosely Connected Task Graphs. *Journal of Computer and System Sciences , August 2003* .	(Start, Date, <b>)	Balasubramaniam, S. and P. Bhardwaj (2004). When not all conflict is bad: Manufacturing marketing conflict and strategic incentive design. *Management Science* 50(4). 489-502.	('.', Volume, <li>)
Devanagari Pen-written Character Recognition. *In ADCOM - 2001* .	(Start, Date, <b>)	Bhardwaj, P. and S. Balasubramaniam (2005). Managing Channel Profits: The Role of Managerial Incentives. Forthcoming *Quantitative Marketing and Economics.*	('.', End, <li>)

Table 2: Two publications pages with different styles. Text in parentheses shows the values of three properties on the correct labelings: (i) Token before Title (ii) First non-Other label after Venue (iii) HTML tag containing Title. The properties are largely uni-modal inside a page, but the mode varies across pages.

Note that 'IEEE' will tend to recur across multiple records, so the last property corresponds to the unigram-based cliques that we modeled thus far. It is clear now that we can gain a lot more if we couple the records together according to 'richer' properties than just unigrams.

Of course the choice of properties is crucial. For all the properties illustrated above, we expect that the labelings inside the domain agree on the property value, without caring for what the value actually is (which varies from domain to domain). This allows us to use the same property on different domains, with varying formatting and authoring styles. Table 2 illustrates this for two publications pages with different styles. It shows three properties that take on largely similar values across records inside a domain, but the dominant value changes across domains. Thus we can reward associativity of these property values using the same symmetric potentials that we have used for unigram cliques.

Now assume that we have an array of such conformance-promoting properties, and a basic MRF trained on some labeled domains. An effective way of deploying this MRF on a previously unseen domain is by labeling the records in the new domain collectively while encouraging the individual labelings to agree on our set of properties. If the properties continue to remain associative in the unseen domain, then collective inference can be expected to correct a significant number of errors. This provides us with an inference-only approach, in contrast to many existing solutions for domain adaptation which require model re-training (Blei et al., 2002; Blitzer et al., 2006; Mann and McCallum, 2007).

We now give an intuitive description of how the introduction of properties causes only minor changes in our collective inference framework. The modified collective inference objective is now given by:

$$\arg\max_{\mathbf{y}^1,\ldots,\mathbf{y}^N} \left(\sum_{k=1}^{N} \sum_{i=1}^{|\mathbf{x}^k|} \phi_i^k(y_i^k, y_{i-1}^k) \right) + \sum_{\mathrm{g}\in\mathcal{G}} C_{\mathrm{g}}(\{\mathrm{g}(\mathbf{x}^k, \mathbf{y}^k)\}_k),$$

instead of Equation 2. Here $\mathcal{G}$ is the set of properties, each of which defines an associative clique. We reuse the cluster message passing setup in this collective inference scenario. We define one cluster per chain, one per property, while separators remain singleton—they correspond to the property value. The new messages are of the kind $m_{\mathrm{g}\to k}(v)$ and $m_{k\to\mathrm{g}}(v)$, where v is a possible value output by the property g(). The property-to-chain messages are computed as before using our potential-specific clique inference algorithms. However the chain-to-property messages $m_{k\to\mathrm{g}}$ cannot be computed for arbitrary properties. To illustrate, say $\mathrm{g}(\mathbf{x}^k, \mathbf{y}^k) =$ Number of tokens marked Author in $\mathbf{y}^k$. Computing chain messages for this property is quite expensive, if not hard. So for computational reasons we restrict ourselves to *Markovian properties*, that is, properties whose chain-to-property messages can be computed with a first-order algorithm like Viterbi or max-product. Examples of Markovian properties include—Token before Title, and Label after Author. A sample message for the second property would be, say, $m_{k \to \mathrm{g}}(\text{Title})$, which can be computed by running Viterbi with the first-order constraint that Author be followed by Title. In general, values of Markovian properties depend only on the Markovian blanket of a part of the chain, which leads to tractability of message computation. We refer the reader to Gupta et al. (2009) for the formal technical details of chain-to-property message computation for Markovian properties.

We stress that such a support for Markovian properties is not possible in alternative collective inference approaches like TRW-S. This is because even with Potts potentials over properties, the inter-cluster separators still remain complete chain labelings instead of scalar property values, so the cluster model cannot be represented as a pairwise graphical model.

In Section 7.3 we will look at the effect of introducing associative properties in the context of domain adaptation on a citation extraction task.

7. Experiments

Our goal is to empirically demonstrate that cluster message passing is indeed a more accurate and efficient framework for doing collective inference. Once this is established, it would necessitate the design of fast and accurate message computation algorithms that work for symmetric clique potentials. This would justify our design and analysis of the various clique inference algorithms presented in this paper. After illustrating the effectiveness of cluster message passing, we will show that the extension of collective inference to capture richer associative properties leads to significant boosts in domain adaptation tasks. Keeping these goals in mind, we present results of three different experiments—clique inference, collective inference, and the extension to general Markovian properties.

First, in Section 7.1 we compare our clique inference algorithms against applicable alternatives in the literature. We compare the algorithms on computation speed and accuracy of the MAP assignments. For Potts potentials, we show that α-pass provides similar MAPs as the various alternatives but is up to two orders faster. For linear MAJORITY potentials (Equation 6), we compare our algorithms against the exact LP-based approach of Section 5.3.2 and the ICM algorithm. For other clique potentials that are not decomposable along the clique edges, we compare α-pass against the ICM algorithm.

Second, in Section 7.2 we show that message passing on the cluster graph is a more effective method for collective inference than its alternatives. For collective models with Potts potentials, we compare cluster message passing against belief propagation that decomposes Potts along the clique edges. In the case of linear MAJORITY potentials, we compare against the stacking based approach of Krishnan and Manning (2006).

Finally, in Section 7.3 we demonstrate the application of our generalized collective framework on domain adaptation. On a citation extraction task, we show that capturing the associativity of richer properties leads to significant reduction in test error.

λ	**ICM**	**α-pass**	**α^2-pass**	**α-exp DP**	**α-exp**	**FastCut**	**FastPD**	**TRW-S**	**DD**
				Clique MAP scores					
0.50	4866.7	4862.3	4864.2	4866.0	4867.0	4867.2	4866.7	4866.4	4833.3
0.55	4878.0	4872.6	4875.5	4878.5	4879.6	4879.7	4879.1	4878.6	4838.1
0.60	4899.8	4893.8	4895.8	4898.0	4900.9	4900.8	4900.7	4899.7	4865.6
0.65	4914.9	4909.4	4911.6	4913.3	4915.9	4916.9	4914.6	4915.1	4881.3
0.70	4919.6	4913.0	4916.7	4916.6	4921.5	4923.1	4920.6	4919.6	4891.2
0.75	4930.0	4934.7	4936.9	4928.8	4935.1	4936.8	4938.2	4938.4	4926.8
0.80	4965.0	4969.3	4972.5	4959.5	4959.5	4971.2	4954.3	4971.4	4961.7
0.85	4977.3	5009.6	5010.0	4995.5	4988.3	5002.8	4997.7	5007.0	4999.9
0.90	5018.5	5082.3	5082.3	5073.9	5081.2	5081.1	5080.1	5079.0	5049.1
0.95	5053.5	5155.9	5155.9	5154.4	5154.9	5155.1	5154.9	5150.7	5100.9
1.00	5137.2	5264.3	5264.3	5264.3	5264.3	5264.3	5264.3	5262.1	5161.9
1.05	5279.4	5417.1	5417.1	5417.1	5417.1	5417.1	5417.1	5417.1	5269.5
1.10	5383.8	5528.1	5528.1	5528.1	5528.1	5528.1	5528.1	5528.1	5358.1
All	65223.5	65812.5	65831.2	65794.0	65813.5	65844.2	65816.6	65833.4	65137.3
				Running Time (ms)					
0.50	24	38	464	155	5900	7910	2510	12170	117180
0.55	27	38	455	174	6610	8010	2930	13290	117480
0.60	28	38	454	160	8140	8810	3640	16290	118270
0.65	29	38	452	157	8700	9120	4190	17740	117050
0.70	35	43	470	162	11680	9630	5210	19480	118360
0.75	34	41	460	155	13770	10460	7180	21730	117170
0.80	42	39	459	153	16710	11060	8650	22660	117450
0.85	38	44	464	139	18260	11310	7360	22610	117730
0.90	42	42	462	163	19100	14980	7280	20170	117670
0.95	49	39	458	127	16450	13640	6620	17910	117070
1.00	58	39	456	85	11280	11200	6320	10670	117240
1.05	59	39	458	82	10730	10800	5810	5280	116780
1.10	50	39	467	82	10490	10420	6000	3400	116920
All	514	519	5979	1795	157820	137350	73700	203400	1526370

Table 3: Clique MAP scores and runtimes of various clique inference algorithms for Potts potential. Each number is an aggregate over 25 cliques for the corresponding λ.

7.1 Clique Inference Experiments

For clique potentials decomposable over clique edges, we compare our clique inference algorithms against sequential tree re-weighted message passing (TRW-S), graph-cut based inference (α-exp), the ICM algorithm, and recent advancements including the faster graph-cut algorithm of Alahari et al. (2008) (FastCut), the fast primal-dual algorithm of Komodakis and Tziritas (2007) (FastPD), and the dual-decomposition scheme of Komodakis et al. (2007a) (DD). For non-decomposable potentials, we present comparisons against the ICM algorithm. We present comparison results on running time and quality of the MAP. Our experiments were performed on both synthetic and real data.

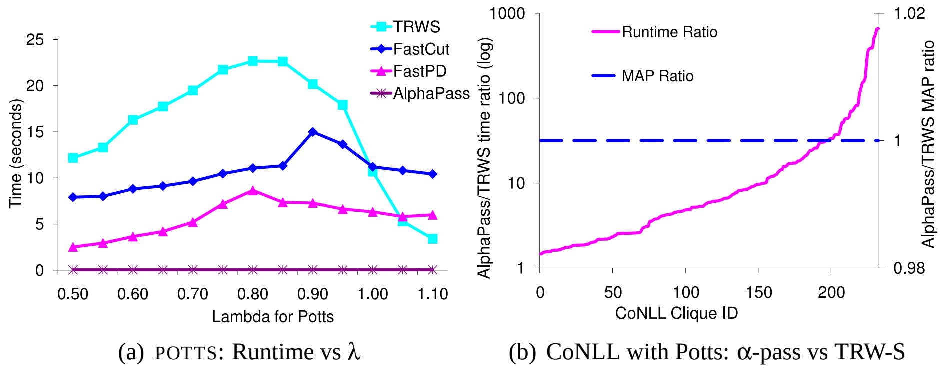


Figure 2: Potts potential: (a) Runtime of α-pass vs TRW-S, Graph-cut and FastPD aggregated on 25 cliques as λ is varied in the synthetic data set POTTS (b) Comparing α-pass and TRW-S on MAP scores and runtimes for each CoNLL clique.

Synthetic Data Set: We generated cliques with 100 vertices and $m = 24$ labels by choosing vertex potentials at random from $[0,2]$ for all vertex-label pairs. A Potts version (POTTS) was created by gradually varying λ, and generating 25 cliques for every value of λ. We also created analogous ENTROPY, MAKESPAN and MAKESPAN2 versions of the data set by choosing entropy ($\lambda \sum_y n_y \log n_y$), linear makespan ($\lambda \max_y n_y$) and square makespan ($\lambda \max_y n_y^2$) clique potentials respectively.

For linear MAJORITY potentials we generated two kinds of data sets (parameterized by λ): (a) MAJ-DENSE obtained by generating a random symmetric W for each clique, where $W_{yy} = \lambda$ was the same for all y and $W_{yy'} \in [0, 2\lambda]$ $(y \neq y')$, and (b) MAJ-SPARSE from symmetric W with $W_{yy'} \in [0, 2\lambda]$ for all y, y', roughly 70% of whose entries were zeroed. This sparse data set is supposed to mirror the sparseness of W in real-life data sets.

CoNLL Data Set: The CoNLL 2003 data set[1] is a popular choice for demonstrating the benefit of collective labeling in named entity recognition tasks. We used the BIOU encoding of the entities, that resulted in 20 labels. We took a subset of 1460 records from the test set of CoNLL, and selected all 233 cliques of size 10 and above. The smaller cliques were ignored as the algorithms hardly differ in performance over them. The median and largest clique sizes were 16 and 259 respectively. The vertex potentials of the cliques were set by a sequential Conditional Random Field trained on a separate training set. We created a Potts version by setting $\lambda = 0.9/n$ using the development set, where n is the clique size. Such a λ allowed us to balance the vertex and clique potentials for each clique. A majority version was also created by learning W discriminatively in the training phase.

We developed Java implementations for all our algorithms—α-pass, generalized α-pass with $q = 2$, dynamic programming based α-expansion of Section 5.2.2 (denoted α-exp DP), modified α-pass, Lagrangian relaxation, and the Exact LP-based algorithm for Majority potentials. We used publicly available C++ implementations for TRW-S,[2] (Boykov et al., 2001; Szeliski et al., 2006; Kolmogorov and Zabih, 2004; Boykov and Kolmogorov, 2004), FastCut (Alahari et al., 2008), and FastPD (Komodakis and Tziritas, 2007; Komodakis et al., 2008). In addition, we implemented DD

1. This data set can be found at `http://cnts.uia.ac.be/conll2003/ner/`.
2. This code can be found at `http://www.adastral.ucl.ac.uk/~vladkolm/papers/TRW-S.html` and `http://vision.middlebury.edu/MRF/`.

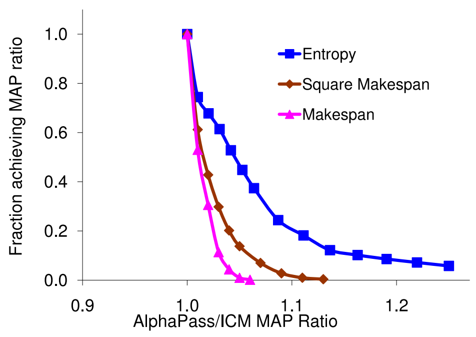


Figure 3: Comparing the MAP quality of α-pass vs ICM on non-decomposable potentials. Plotted point (r, f) denotes that for a fraction f of the cliques, α-pass returned a MAP score at least r times that of ICM.

in C++ and ICM in Java. All experiments were performed on a Pentium-IV 3.0 GHz machine with 8 GB of RAM.

7.1.1 EDGE DECOMPOSABLE POTENTIALS

Table 3 compares the various clique inference algorithms on the POTTS data set. We vary the Potts parameter λ uniformly in the range $[0.5, 1.1]$. This range is of special interest because it allows maximal contention between the clique and vertex potentials. For λ outside this range, the MAP is usually a trivial assignment, viz. one which either individually assigns each vertex to its best label (optimizing vertex potential), or assigns all vertices to a single label (optimizing clique potential). Each number in the upper half of Table 3 is the aggregate MAP score of 25 cliques for that particular λ, while the bottom half reports the aggregate clique inference time rounded to the nearest millisecond.

We observe that apart from ICM and DD, all the other algorithms return MAP scores in a small interval (0.5%) of each other. In particular α-pass performs almost identical to α-exp and FastPD in this regard. FastCut provides the best aggregate MAP scores, but α^2-pass and TRW-S are quite close as well. We note that the three expansion algorithms α-exp, α-exp DP, and FastCut return different MAP scores, primarily because they employ different bootstrapping steps. Further we observe that DD returns significantly low MAP scores, because even with suitable step-size selection heuristics such as the one proposed by Komodakis et al. (2007a), DD converges very slowly and invariably hits the maximum iteration limit of 30 with a sub-par MAP. ICM also returns lower MAP scores, as expected, due to its highly local updates.

In terms of runtime, Table 3 shows a clear separation of the algorithms. ICM and α-pass are the fastest, while α-exp, FastCut, FastPD, and TRW-S are 150-400 times slower. This is because in contrast to α-pass, the other algorithms perform multiple costly iterations (e.g., an iteration of TRW-S is $O(n^2)$). The cut-based algorithms and FastPD are faster than TRW-S, while DD is the slowest, at 3000 times α-pass. This is because DD has to deal with $O(n)$ spanning trees which is costly, and leads to slow convergence. Consequently DD hits the iteration limit every time. Our α^2-pass and dynamic programming based α-expansion (α-exp DP) algorithms are ten and three times slower than α-pass respectively, but still more than an order faster than the other algorithms. Finally we see that while α-pass and α^2-pass take almost a constant time irrespectively of λ, the other schemes take up more time (as they iterate more) around the "maximal contention range" of $[0.7, 1.0]$ for λ. Figure 2(a) illustrates this behavior, where we vary λ and plot the runtimes of α-pass, FastPD, TRW-S, and FastCut. Note that α-pass lies very close to the x-axis, illustrating its small runtime.

Figure 2(b) presents the results on Potts cliques from the CoNLL data set. For simplicity we only compare α-pass with TRW-S as the other algorithms behave analogous to the synthetic scenario. For each clique, we plot (a) the ratio of the α-pass MAP score with that of TRW-S, and (b) ratio of TRW-S runtime vs α-pass runtime. While both the algorithms report the same MAP, α-pass is still more than 10 times faster on more than one-third of the cliques, and is never slower. This ratio is not as bad as for synthetic cliques mainly because the median clique size here is much smaller at 16.

7.1.2 NON-DECOMPOSABLE POTENTIALS

In this case, we cannot compare against the TRW-S or graph-cut based algorithms. Hence we compare with the ICM algorithm that has been popular in such scenarios (Lu and Getoor, 2003). The scheme of Potetz and Lee (2008) is another alternative, but we omit a comparison with it as it is bound to be quite expensive in m (see discussion in Section 8).

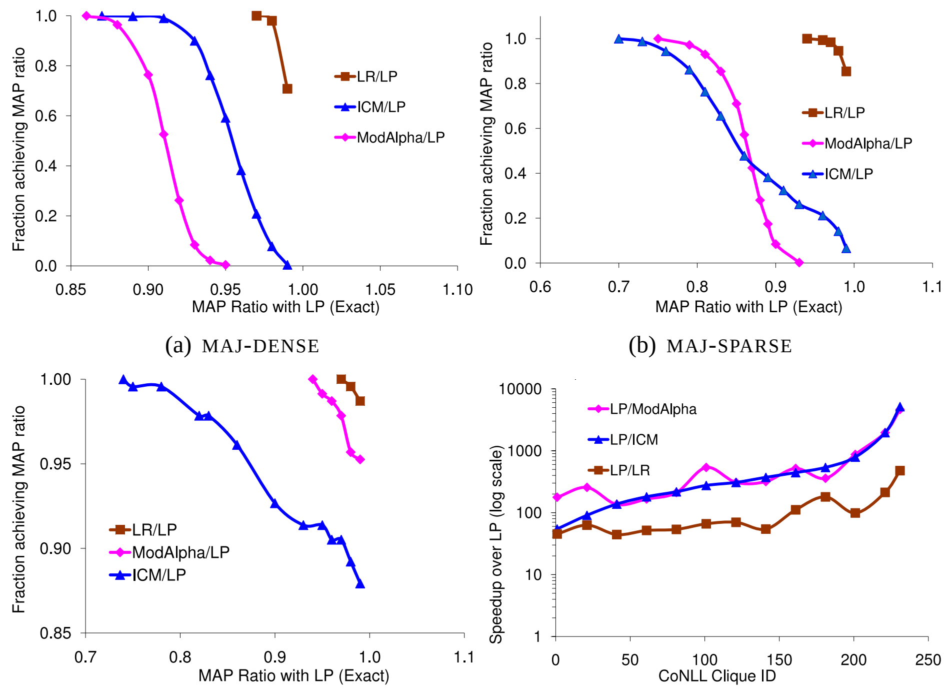
We varied λ with increments of 0.02 in $[0.7, 1.1)$ and generated 500 cliques each from MAJ-DENSE, MAJ-SPARSE, ENTROPY, MAKESPAN and MAKESPAN2. We measure the ratio of MAP score of α-pass with ICM and for each ratio r we plot the fraction of cliques where α-pass returns a MAP score at least r times that of ICM. Figure 3 shows the results on all the potentials except MAJORITY, which will be presented later. The curves for linear and square makespan lie totally to the right of $ratio = 1$, which is expected because α-pass will always return the true MAP for those potentials. In contrast ICM can only return a locally optimal solution. For entropy, α-pass was found to be significantly better than ICM in all the cases. The runtimes of ICM and α-pass were similar.

7.1.3 MAJORITY POTENTIALS

In Figures 4(a) and 4(b), we compare ICM, Lagrangian Relaxation (LR) and modified-α-pass (Section 5.3.1, denoted ModAlpha) against the LP-based exact method (LP) on synthetic data. Each curve plots, for each MAP ratio r, the fraction of cliques on which ICM (or LR or ModAlpha) returns a MAP score better than r times the optimal MAP score. An ideal algorithm's curve would just be a dot at $(1, 1)$ indicating that it retrieves the true MAP for all the cliques.

We observe that on MAJ-DENSE, both ModAlpha and ICM return a MAP score better than 0.85 of the true MAP, with ICM being slightly better. However, LR out-performs both of them, providing a MAP ratio always better than 0.97 and returning the true MAP in more than 70% of the cases. In MAJ-SPARSE too, LR dominates the other two algorithms, returning the true MAP in more than 80% of the cases, with a MAP ratio always better than 0.92. Further it can be derived that on average, LR returns a MAP score 1.15 times that of ICM. Thus, LR performs much better than its competitors across dense as well as sparse majority potentials.

The results on CoNLL data set, whose W matrix is 85% sparse, are displayed in Figure 4(c). ICM, ModAlpha, and LR return the true MAP in 87%, 95% and 99% of the cliques respectively, with the worst case MAP ratio of LR being 0.97 as opposed to 0.94 and 0.74 for ModAlpha and ICM respectively. Figure 4(d) displays runtime ratios on all CoNLL cliques for all three inexact algorithms vs LP. ICM and ModAlpha are roughly 100-10000 times faster than LP, while LR is



(a) MAJ-DENSE (b) MAJ-SPARSE (c) CoNLL (d) Time(CoNLL)

Figure 4: (a)-(c) MAP quality of modified α-pass, ICM, LR vs LP on MAJORITY potentials over MAJ-DENSE, MAJ-SPARSE, and CoNLL. To be interpreted in the same way as Figure 3. (d) Runtime ratios vs LP of these algorithms for each CoNLL clique.

only 3 – 5 times more expensive than ICM and ModAlpha on average. Thus, for practical majority potentials, LR and ModAlpha quickly provide highly accurate solutions.

7.2 Collective Labeling of Repeated Words

We now establish that for collective inference setups like the one in Figure 1(d), message passing on the cluster graph (denoted CI) is a better option than the alternatives. This would justify the design of special clique inference algorithms such as α-pass and LR.

We consider information extraction over text records, and define cliques over repeated occurrences of words. We create two versions of the experiment—with Potts and MAJORITY potentials on the cliques respectively. Message computation at those cliques will be done using α-pass and LR respectively, as we have already established their efficiency and accuracy in Section 7.1.

For the edge-decomposable Potts version, we compare CI against TRW-S on the giant pairwise graph. Other algorithms such as FastPD, FastCut, and α-exp are not applicable as none of the chain edge potentials in the giant graph are semi-metric or submodular. For the MAJORITY version, we compare CI against the stacking approach of Krishnan and Manning (2006).

We report results on three data sets—the Address data set consisting of roughly 400 non-US postal addresses, the Cora data set (McCallum et al., 2000) containing 500 bibliographic records, and the CoNLL’03 data set. The training splits were 30%, 10% and 100% respectively for the three data sets, and the parameter λ for Potts was set to 0.2,1 and 0.05 using cross-validation on a held out split. The MAJORITY parameter W was learnt generatively through label co-occurrence statistics in the cliques present in the training data. We report the token-F1 scores as a measure of accuracy of the various approaches.

Figure 5 reports the combined token-F1 over all labels except 'Other'. Unless specified otherwise, all the approaches post statistically significant gains over the base model. The accuracies show only modest improvements over the base model. This is because our cliques are of a highly limited form and so we cannot expect to correct too many errors using a collective model. We will look at more complex cliques in Section 7.3. Coming back to Figure 5, for MAJORITY potentials, CI is superior to the stacking based approach. The difference is statistically significant for Cora and CoNLL'03. For the Potts version, TRW-S and CI provide similar gains over Address and Cora. We could not run TRW-S on CoNLL'03, as the resulting graph was too big for the TRW-S code to handle.

We now compare the different approaches on running time. In Figure 6 we plot the accuracy of the two methods versus the number of iterations. CI achieves its best accuracy after just one round of message passing, whereas TRW-S takes around 20 iterations. In terms of clock time, an iteration of TRW-S costs $\sim$ 3.2s for CORA, and that of CI costs 3s, so CI is roughly an order of magnitude faster than TRW-S for the same accuracy levels. The comparison was similar for the Address data set.

Potential	**Model**	**Addr**	**Cora**	**CoNLL**
	Base	81.5	88.9	87.0
Potts	CI	81.9	89.7	88.8
	TRW-S	81.9	89.7	-
Majority	CI	82.2	89.6	88.8
	Stacking	81.7*	87.5$\downarrow$	87.8

Figure 5: Token-F1 of various collective inference schemes. F1 averaged over five splits for Address and Cora. '*' and $\downarrow$ denote statistically insignificant difference and significant loss over Base respectively.

7.3 Domain Adaptation

We move on to a generalization of our collective inference framework, and show that capturing associativity of a richer set of properties can help us in domain adaptation. We focus on a citation extraction task, where the aim is to adapt a sequential model across widely varying publications pages of authors. Our data set consists of 433 bibliographic entries from the web-pages of 31 authors, hand-labeled with 14 labels such as Title, Author, Venue, Location and Year. Bibliographic entries across different authors differ in many aspects like label-ordering, missing labels, punctuation, HTML formatting and bibliographic style.

A fraction of the 31 domains were used to train a baseline sequential model. The model was trained with the LARank algorithm of Bordes et al. (2007), using the BCE encoding for the labels.

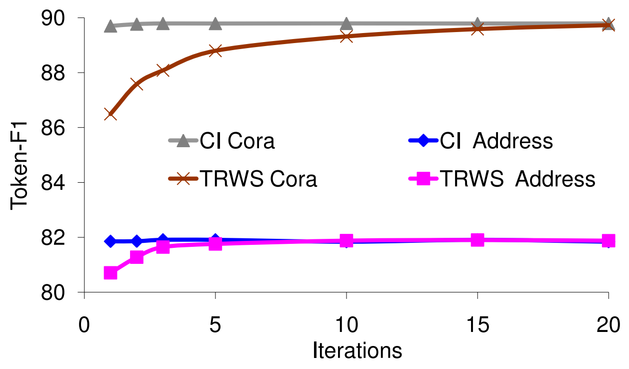


Figure 6: F1 vs iterations for CI vs TRW-S on Cora and Address.

We used standard extraction features in a window around each token, along with label transition features (Peng and McCallum, 2004).

For our collective framework, we used the following Markovian properties:

$$\begin{aligned}
g_1(\mathbf{x},\mathbf{y}) &= \text{First non-Other label in } \mathbf{y} \\
g_2(\mathbf{x},\mathbf{y}) &= \text{Token before the Title segment in } \mathbf{y} \\
g_3(\mathbf{x},\mathbf{y}) &= \text{First non-Other label after Title in } \mathbf{y} \\
g_4(\mathbf{x},\mathbf{y}) &= \text{First non-Other label after Venue in } \mathbf{y}
\end{aligned}$$

Inside a domain, any one of the above properties will predominantly favor one value, for example, g_3 might favor the value ‘Author’ in one domain, and ‘Date’ in another. Thus these properties encourage consistent labeling around the Title and Venue segments. We use Potts potential for each property, setting $\lambda = 1$ using cross-validation.

We reiterate that there are no alternative collective inference schemes for this property-based framework. This is primarily because other algorithms like TRW-S or ordinary belief propagation cannot deal with property-based separators (ref. Section 6).

The performance results of CI with the above properties versus the baseline model are presented in Figure 7. For the test domains, we report token-F1 of the important labels—Title, Author and Venue. The accuracies are averaged over five trials. CI leads to upto 25% reduction over the base test error for Venue and Title, labels for which we had defined related properties. The gain is statistically significant ($p < 0.05$). Though the improvement is more prominent when only a few domains are available for training, we continue to see an improvement even with more training domains as there invariably are new styles in the test data. Figure 8 shows the error reduction on individual test domains for one particular train-test split of five and 26 domains respectively. The errors are computed from the combined token F1 scores of Title, Venue and Author. For some domains the errors are reduced by more than 50%. Collective inference increases errors in only two domains. Such an increase happens when most of the records in the domain take on wrong property values, so collective inference ends up reinforcing those errors by wrongly biasing the remaining minority of the records that have correct property values.

Finally, we mention that for this task, applying the classical collective inference setup with cliques over word repetitions leads to very minor gains. This is because most of the word cliques already agree on their vertex labels under the base model, so collective inference with word cliques does not add too much value. In this context, the generalized collective inference framework is indeed a much more accurate mechanism for joint labeling.

Train	**Title**		**Venue**		**Author**	
(%)	Base	CI	Base	CI	Base	CI
5	70.7	74.8	58.8	62.5	74.1	74.3
10	78.0	82.1	69.2	72.2	75.6	75.9
20	85.8	88.6	76.7	78.9	80.7	80.7
30	91.7	93.0	81.5	82.6	87.7	88.0
50	92.3	94.2	83.5	84.5	89.4	90.0

Figure 7: Token-F1 of CI and Base

8. Related Work

We group the known approaches into various categories and compare them with our collective inference framework.

8.1 Generic Collective Inference Approaches

Collective graphical models have been used to capture associativity in many text mining tasks such as IE, entity labeling, and document classification (Sutton and McCallum, 2004; Finkel et al., 2005; Bunescu and Mooney, 2004; Krishnan and Manning, 2006; Kulkarni et al., 2009; Chakrabarti et al., 1998; Lu and Getoor, 2003; Taskar et al., 2004). However these models use generic algorithms like ordinary belief propagation (Sutton and McCallum, 2004; Bunescu and Mooney, 2004; Taskar et al., 2004), Gibbs sampling (Finkel et al., 2005), local search (Lu and Getoor, 2003) or multi-stage schemes (Krishnan and Manning, 2006). Our framework is general enough to support various clique potentials, yet exploits the structure of the potential to efficiently compute a full set of messages for collective inference.

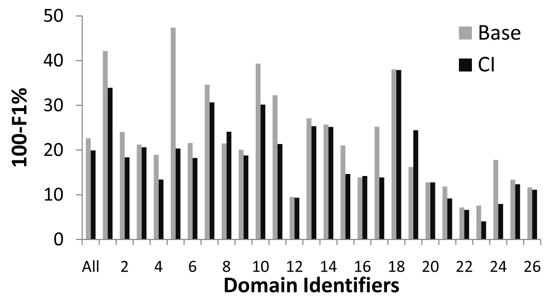


Figure 8: Per-domain F1-error

8.2 MAP Inference on Pairwise Models

We provide only a brief overview of the many recent advances in inference over pairwise graphs since our main interest lies in higher-order potentials. These approaches fall under two broad categories. The first category has message passing algorithms that solve the dual of an LP relaxation, including TRW-S (Kolmogorov, 2006), max-sum diffusion (Werner, 2009), and other convergent alternatives (Meltzer et al., 2009; Ravikumar et al., 2010). These LP relaxations, that only impose local pairwise constraints, have been tightened in various ways using cycle and higher order marginal constraints (Sontag et al., 2008; Werner, 2009; Komodakis and Paragios, 2008; Kumar et al., 2009). The second category includes combinatorial algorithms based on graph-cuts. For binary labels, two well known methods are the graph-cut method for submodular potentials (Boykov et al., 2001) and the Quadratic Pseudo-Boolean Optimization method (QPBO) for getting a partial solution for arbitrary potentials (Boros and Hammer, 2002; Kovtun, 2003; Kolmogorov and Rother, 2007). For multi-label models with metric edge potentials, the α-expansion algorithm of Boykov et al. (2001) provides a 1/2-approximation. α-expansion has subsequently been generalized, analyzed and optimized (Veksler, 2007; Kumar and Torr, 2008b; Lempitsky et al., 2007; Komodakis and Tziritas, 2005; Komodakis et al., 2007b; Alahari et al., 2008). In particular, the FastPD algorithm of Komodakis et al. (2007b) is a primal-dual generalization which works even with semi-metric edge potentials. However our chain edge potentials are not even semi-metric. Similarly the partial optimality guarantee of QPBO is of limited appeal, as our model has a large fraction of non-submodular edges.

8.3 Alternate Collective Inference Frameworks

Recently, two inference frameworks, the LP relaxation of Werner (2008) and the dual-decomposition framework of Komodakis and Paragios (2009) have been extended to handle higher-order potentials. In the LP relaxation framework, a max-sum diffusion algorithm is used to solve the dual and each step of the algorithm requires the computation of max-marginals from the higher-order cliques. In the dual-decomposition framework, the model is decomposed into tractable components and the component MAPs or marginals are used to construct a sub-gradient of the collective inference LP. Like cluster message passing, these frameworks also allow the plugging-in of arbitrary algorithms for computing max-marginals at each clique. Thus, our clique inference algorithms can be used unchanged in these two frameworks. However, while these alternative frameworks provide interesting new ways of looking at inference, they do not necessarily provide faster convergence guarantees. For example, as noted in Werner (2009), max-sum diffusion converges very slowly as compared to TRW-S even for binary potentials. We showed in Section 7.1 that dual decomposition is significantly slower than α-pass when used for clique inference.

8.4 Transforming Higher-Order Cliques to Pairwise Potentials

A lot of recent work on higher-order clique potentials in vision deals with reducing these potentials to pairwise and then applying α-expansion or QPBO as applicable (Ishikawa, 2009; Kohli et al., 2008; Ramalingam et al., 2008; Rother et al., 2009; Kohli and Kumar, 2010). These transformations add auxiliary nodes, which are few in number when the potentials are truncated and sparse (Rother et al., 2009). However our potentials are dense and not truncated. Consequently these techniques will introduce too many extra nodes, for example, exponential in n (Ishikawa, 2009), or $O(mn)$

(Kohli et al., 2008). Similarly, the method of Kohli and Kumar (2010) will expand the node set by a factor equal to the number of functions used to approximate our potential (and this can be quite large). Too many auxiliary nodes makes inference using α-expansion/QPBO quite expensive. In contrast we perform clique inference without any transformation, and our clique inference bounds are better than the alternatives.

We also note that there are transformations that extend α-expansion to metric potentials over triangles (Kolmogorov and Zabih, 2004), and truncated versions of Potts potentials over arbitrarily large cliques (Kohli et al., 2007, 2009). Our symmetric potentials are not truncated so the same transformations cannot be applied to our model.

8.5 Special Higher-Order Potentials

An orthogonal body of work deals with special higher-order potentials that allow efficient message computation, such as linear-constrained potentials (Potetz and Lee, 2008), convex and order-based potentials (Tarlow et al., 2010), and decomposable potentials (Chen et al., 2008). Although our cluster graph can seamlessly incorporate these potentials and their clique inference algorithms, it is fruitful to compare some of these potentials with ours. Linear-constrained potentials assume that the labels are numeric and the potential's value depends only on a linear combination of node labels. Such potentials allow $O(nm^2)$ message computation using clever variable substitution during belief propagation. When we have only two labels, our symmetric potentials can be encoded as linear-constrained potentials, thus leading to $O(n)$ clique inference as compared to $O(n \log n)$ using α-pass. However, this encoding does not have an inexpensive generalization to *discrete* multi-label cliques. Also, representing multi-label potentials using a collection of linear-constrained potentials can make clique inference exponential in m (Potetz and Lee, 2008). Chen et al. (2008) propose *decomposable potentials* that are sums of a few sub-potentials, each of which is a product of low-arity functions. This includes the *Voting Potential*, which combines associativity scores additively along edges unlike multiplicatively as in Potts.[3] For some decomposable potentials, including the Voting Potential, exact messages can be computed in closed form.

Symmetric potentials have been used in special ways in other inference tasks too. Jaimovich et al. (2007) proposed a framework for inference over relational data where symmetric potentials are used to collapse large instance-level factor graphs into template-level graphs. Similarly, Milch et al. (2008) proposes lifting techniques with counting potentials. In our framework we have a mix of symmetric potentials, individual node and non-symmetric edge potentials. We therefore cannot perform any kind of collapsing or lifted inference.

9. Conclusions and Future Work

We presented a general collective inference framework that exploits the associativity of a rich set of properties of instances and their labelings. We argued that cluster message passing, which exploits the special associative structure and computes a complete set of messages, is a more principled inference mechanism than other cluster-oblivious or message-oblivious approaches. We demonstrated the effectiveness of the framework on a real-life domain adaptation task.

3. Although Potts decomposes additively along the clique edges, the probability $P(\mathbf{y}|\mathbf{x})$ is exponential in the clique potential term, which makes it multiplicative.

We presented potential-specific combinatorial algorithms for message computation in associative cliques. We presented the α-pass algorithm which is sub-quadratic in the clique size and gives the exact MAP for all MAX clique potentials, and any symmetric potential with two labels, and a tight approximation guarantee of $\frac{13}{15}$ on the Potts potential. We showed that α-pass is significantly faster while providing the same or better accuracy than alternatives such as TRW-S, graph-cuts, and their recently proposed improved versions. We gave a Lagrangian relaxation method for generating messages from a clique with majority potential. This algorithm is at least two orders of magnitude faster than an exact algorithm and more accurate than other approximate approaches.

Our future work includes automated property induction to figure out rich associative properties in unlabeled domains. We are also interested in applying symmetric potentials to general dense subgraphs instead of cliques. We believe that this might help in semi-supervised learning tasks.

Appendix A. Proofs

Theorem 7 $F(\hat{\mathbf{y}}) \geq \frac{13}{15}F(\mathbf{y}^*)$. *Further, this ratio is tight.*

Proof The proof is by contradiction. Suppose there is an instance where $F(\hat{\mathbf{y}}) < \frac{13}{15}F(\mathbf{y}^*)$. Wlog assume that $\lambda = 1$ and $n_1 \geq n_2 \geq \ldots \geq n_k > 0$, $(2 \leq k \leq m)$ be the non-zero counts in the optimal solution and let ψ^* be its vertex score. Thus $F(\mathbf{y}^*) = \psi^* + n_1^2 + n_2^2 + \ldots + n_k^2$.

Now, $F(\hat{\mathbf{y}})$ is at least $\psi^* + n_1^2$ (ref. Claim 5.1). This implies $\frac{\psi^*+n_1^2}{\psi^*+n_1^2+\ldots+n_k^2} \leq \frac{13}{15}$, that is, $2(\psi^* + n_1^2) < 13(n_2^2 + \ldots + n_k^2)$ or

$$\psi^* < \frac{13}{2}(n_2^2 + \ldots + n_k^2) - n_1^2. \tag{18}$$

Since k labels have non-zero counts, and the vertex score is ψ^*, at least ψ^*/k of the vertex score is assigned to one label. Considering a solution where all vertices are assigned to this label, we get $F(\hat{\mathbf{y}}) \geq \psi^*/k + n^2$.

Therefore $F(\mathbf{y}^*) > 15/13(n^2 + \psi^*/k)$.

Since $F(\mathbf{y}^*) = \psi^* + n_1^2 + \ldots + n_k^2$, we get:

$$\psi^* > \frac{15kn^2 - 13k(n_1^2 + \ldots + n_k^2)}{13k - 15}. \tag{19}$$

We show that Equations 18 and 19 contradict each other. It is sufficient to show that for all $n_1 \geq \ldots \geq n_k \geq 1$,

$$\frac{15kn^2 - 13k(n_1^2 + \ldots n_k^2)}{13k - 15} \geq \frac{13}{2}(n_2^2 + \ldots + n_k^2) - n_1^2.$$

Simplifying, this is equivalent to

$$kn^2 - \frac{13}{2}(k-1)(n_2^2 + \ldots + n_k^2) - n_1^2 \geq 0.$$

Consider a sequence $n_1, \ldots, n_k$ for which the expression on the left hand side is minimized. If $n_i > n_{i+1}$ then we must have $n_l = 1\ \forall l \geq i+2$. Otherwise, replace n_{i+1} by $n_{i+1}+1$ and decrement n_j by 1, where j is the largest index for which $n_j > 1$. This gives a new sequence for which the value of the expression is smaller. Therefore the sequence must be of the form $n_i = n_1$ for $1 \leq i < l$ and $n_i = 1$ for $i > l$, for some $l \geq 2$. Further, considering the expression as a function of n_l, it is quadratic with a negative second derivative. So the minimum occurs at one of the extreme values $n_l = 1$ or $n_l = n_1$. Therefore we only need to consider sequences of the form $n_1, \ldots, n_1, 1, \ldots, 1$ and show that the expression is non-negative for these.

In such sequences, differentiating with respect to n_1, the derivative is positive for $n_1 \geq 1$, which means that the expression is minimized for the sequence $1, \ldots, 1$. Now it is easy to verify that it is true for such sequences. The expression is zero only for the sequence $1, 1, 1$, which gives the worst case example.

We now illustrate the tightness of this ratio through a pathological instance where the solution of α-pass is exactly $\frac{13}{15}$ of the optimal. The clique is constructed as follows. Let $m = n+3$ and $\lambda = 1$. For the first $n/3$ vertices let $\psi_{u1} = 4n/3$, for the next $n/3$ vertices let $\psi_{u2} = 4n/3$, and for the remaining $n/3$ let $\psi_{u3} = 4n/3$. Also for all vertices let $\psi_{u(u+3)} = 4n/3$. All other vertex scores are zero. The optimal solution is to assign the first three labels $n/3$ vertices each, yielding a score of $4n^2/3 + 3(\frac{n}{3})^2 = 5n^2/3$. The first α-pass with $\alpha = 1$, where initially a vertex u is assigned its vertex optimal label $u+3$, will assign the first $n/3$ vertices label 1. This keeps the sum of total vertex scores unchanged at $4n^2/3$, the clique score increases to $n^2/9 + 2n/3$ and total score = $4n^2/3 + n^2/9 + 2n/3 = 13n^2/9 + 2n/3$. No subsequent combinations with any other label α can improve this score. Thus, the score of α-pass is $\frac{13}{15}$ of the optimal in the limit $n \to \infty$. ■

Theorem 8 *Generalized α-pass enjoys an approximation bound of $\frac{8}{9}$, that is, $F(\hat{\mathbf{y}}) \geq \frac{8}{9} F(\mathbf{y}^*)$.*

Proof This bound is achieved if we run the algorithm with $q = 2$. Let the optimal solution have counts $n_1 \geq n_2 \geq \ldots \geq n_m$ and let its vertex score be ψ^*. For simplicity let $a = n_1/n$, $b = n_2/n$ and $c = \psi^*/n^2$. Then $F(\mathbf{y}^*)/n^2 \leq c + a^2 + b(1-a)$, $F(\hat{\mathbf{y}})/n^2 \geq c + a^2$ and $F(\hat{\mathbf{y}})/n^2 \geq c + \frac{(a+b)^2}{2}$.
Case 1: $a^2 \geq \frac{(a+b)^2}{2}$. Then $F(\mathbf{y}^*) - F(\hat{\mathbf{y}}) \leq bn^2(1-a)$. For a given value of a, this is maximized when b is as large as possible. For Case 1 to hold, the largest possible value of b is given by $a^2 = \frac{(a+b)^2}{2}$, which gives $b = a(\sqrt{2}-1)$. Therefore $F(\mathbf{y}^*) - F(\hat{\mathbf{y}}) \leq \frac{n^2(\sqrt{2}-1)}{4} < \frac{n^2}{8} \leq \frac{F(\hat{\mathbf{y}})}{8}$, that is, $F(\hat{\mathbf{y}}) \geq \frac{8}{9} F(\mathbf{y}^*)$.
Case 2: $a^2 \leq \frac{(a+b)^2}{2}$. This holds if $b \geq (\sqrt{2}-1)a$. Since $a + b \leq 1$, this is possible only if $a \leq 1/\sqrt{2}$. Now $\frac{F(\mathbf{y}^*) - F(\hat{\mathbf{y}})}{n^2} \leq a^2 + b(1-a) - (a+b)^2/2 = \frac{a^2 - 4ab + 2b - b^2}{2}$.

For a given a, this expression is quadratic in b with a negative second derivative. This is maximized (by differentiating) for $b = 1 - 2a$. Since $b \leq a$, this value is possible only if $a \geq 1/3$. Similarly, for case 2 to hold with this value of b, we must have $a \leq \sqrt{2} - 1$. Substituting this value of b, the difference in scores is $\frac{5a^2 - 4a + 1}{2}$.

Since this is quadratic with a positive second derivative, it is maximized when a has either the minimum or maximum possible value. For $a = 1/3$ this value is $1/9$, while for $a = \sqrt{2} - 1$, it is $10 - 7\sqrt{2}$. In both cases, it is less than $1/8$.

If $a \leq 1/3$ the maximum is achieved when $b = a$. In this case, the score difference is at most $(a - 2a^2)$ which is maximized for $a = 1/4$, where the value is $1/8$. (This is the worst case).

For $\sqrt{2} - 1 < a \leq 1/\sqrt{2}$, the maximum will occur for $b = (\sqrt{2}-1)a$. Substituting this value for b, the score difference is $(\sqrt{2}-1)(a - a^2)$, which is maximized for $a = 1/2$, where its value is $(\sqrt{2}-1)/4 < 1/8$. ■

References

Karteek Alahari, Pushmeet Kohli, and Philip H. S. Torr. Reduce, reuse & recycle: efficiently solving multi-label mrfs. In *CVPR*, 2008.

David Blei, Drew Bagnell, and Andrew McCallum. Learning with scope, with application to information extraction and classification. In *UAI*, 2002.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, 2006.

Antoine Bordes, Léon Bottou, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with larank. In *ICML*, pages 89–96, 2007.

Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 26, no. 9, pages 1124–1137, 2004.

Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

Razvan Bunescu and Raymond J. Mooney. Collective information extraction with relational markov networks. In *ACL*, 2004.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM*, 2010.

Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. *SIGMOD Rec.*, 27(2):307–318, 1998.

Shann-Ching Chen, Geoffrey J. Gordon, and Robert F. Murphy. Graphical models for structured classification, with an application to interpreting images of protein subcellular location patterns. *JMLR*, 9:651–682, 2008. ISSN 1533-7928.

John Duchi, Daniel Tarlow, Gal Elidan, and Daphne Koller. Using combinatorial optimization within max-product belief propagation. In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2007.

Hazem Elmeleegy, Jayant Madhavan, and Alon Halevy. Harvesting relational tables from lists on the web. In *VLDB*, 2009.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

Rahul Gupta and Sunita Sarawagi. Answering table augmentation queries from unstructured lists on the web. In *PVLDB*, 2009.

Rahul Gupta, Ajit A. Diwan, and Sunita Sarawagi. Efficient inference with cardinality-based clique potentials. In *Proceedings of the 24th International Conference on Machine Learning (ICML), USA*, 2007.

Rahul Gupta, Sunita Sarawagi, and Ajit A. Diwan. Generalized collective inference with symmetric clique potentials, 2009. URL `http://arxiv.org/abs/0907.0589v2`.

Hiroshi Ishikawa. Higher-order clique reduction in binary graph cut. In *CVPR*, pages 2993–3000, 2009.

Ariel Jaimovich, Ofer Meshi, and Nir Friedman. Template based inference in symmetric relational markov random fields. In *UAI*, 2007.

Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639, 2002. doi: http://doi.acm.org/10.1145/585265.585268.

Pushmeet Kohli and M. Pawan Kumar. Energy minimization for linear envelope mrfs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. P3 & beyond: Solving energies with higher order cliques. In *CVPR*, 2007.

Pushmeet Kohli, Alexander Shekhovtsov, Carsten Rother, Vladimir Kolmogorov, and Philip H. S. Torr. On partial optimality in multi-label mrfs. In *ICML*, pages 480–487, 2008.

Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006. ISSN 0162-8828.

Vladimir Kolmogorov and Carsten Rother. Minimizing nonsubmodular functions with graph cuts – a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1274–1279, 2007.

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

Nikos Komodakis and Nikos Paragios. Beyond loose lp-relaxations: optimizing mrfs by repairing cycles. In *ECCV*, 2008.

Nikos Komodakis and Nikos Paragios. Beyond pairwise energies: efficient optimization for higher-order mrfs. In *CVPR*, 2009.

Nikos Komodakis and Georgios Tziritas. A new framework for approximate labeling via graph cuts. In *ICCV*, pages 1018–1025, 2005.

Nikos Komodakis and Georgios Tziritas. Approximate labeling via graph cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436–1453, 2007.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. Mrf optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007a.

Nikos Komodakis, Georgios Tziritas, and Nikos Paragios. Fast, approximately optimal solutions for single and dynamic mrfs. In *CVPR*, 2007b.

Nikos Komodakis, Georgios Tziritas, and Nikos Paragios. Performance vs computational efficiency for optimizing single and dynamic mrfs: Setting the state of the art with primal-dual strategies. *Comput. Vis. Image Underst.*, 112(1):14–29, 2008.

Ivan Kovtun. Partial optimal labeling search for a np-hard subclass of (max, +) problems. In *DAGM-Symposium*, pages 402–409, 2003.

Vijay Krishnan and Christopher D. Manning. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *ACL-COLING*, 2006.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *SIGKDD*, 2009.

M. Pawan Kumar and Philip H. S. Torr. Efficiently solving convex relaxations for map estimation. In *ICML*, 2008a.

M. Pawan Kumar and Philip H. S. Torr. Improved moves for truncated convex models. In *NIPS*, pages 889–896, 2008b.

M. Pawan Kumar, Vladimir Kolmogorov, and Philip H. S. Torr. An analysis of convex relaxations for map estimation of discrete mrfs. *JMLR*, 10:71–106, 2009.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.

Victor S. Lempitsky, Carsten Rother, and Andrew Blake. Logcut - efficient graph cut optimization for markov random fields. In *ICCV*, pages 1–8, 2007.

Qing Lu and Lise Getoor. Link-based classification. In *Machine Learning, ICML*, pages 496–503, 2003.

Gideon Mann and Andrew McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *ICML*, 2007.

Andrew McCallum, Kamal Nigam, Jason Reed, Jason Rennie, and Kristie Seymore. Cora: computer science research paper search engine. http://cora.whizbang.com/, 2000.

Talya Meltzer, Amir Globerson, and Yair Weiss. Convergent message passing algorithms - a unifying view. In *UAI*, 2009.

Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted probabilistic inference with counting formulas. In *AAAI*, 2008.

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, chapter 11, pages 247–254. Prentice Hall, Englewood Cliffs, NJ., 1982.

Fuchun Peng and Andrew McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL*, pages 329–336, 2004.

Brian Potetz and Tai Sing Lee. Efficient belief propagation for higher-order cliques using linear constraint nodes. *Computer Vision and Image Understanding*, 112(1):39–54, 2008.

Srikumar Ramalingam, Pushmeet Kohli, Karteek Alahari, and Philip H. S. Torr. Exact inference in multi-label crfs with higher order cliques. In *CVPR*, 2008.

Pradeep 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.

Carsten Rother, Pushmeet Kohli, Wei Feng, and Jiaya Jia. Minimizing sparse higher order energy functions of discrete variables. In *CVPR*, 2009.

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening lp relaxations for map using message passing. In *UAI*, 2008.

Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, 2004.

Rick Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields. In *ECCV 2006*, volume 2, pages 16–29, 2006.

Daniel Tarlow, Inmar Givoni, and Richard Zemel. Hop-map: efficient message passing with high order potentials. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AI-STATS)*, volume 9, pages 812–819. JMLR: W&CP, 2010.

Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative markov networks. In *Twenty First International Conference on Machine Learning (ICML04)*, 2004.

Olga Veksler. Graph cut based optimization for mrfs with truncated convex priors. In *CVPR*, 2007.

Tomás Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (map-mrf). In *CVPR*, 2008.

Tomas Werner. Revisiting the linear programming relaxation approach to gibbs energy minimization and weighted constraint satisfaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2009. ISSN 0162-8828.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring Artificial Intelligence in the New Millenium*, 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1-55860-811-7.