Approximate Inference Using DC Programming For Collective Graphical Models

Duc Thien Nguyen[†] **Akshat Kumar**[†] **Hoong Chuin Lau**[†] **Daniel Sheldon**[‡]
[†]Singapore Management University, [‡]University of Massachusetts Amherst
[†]{dtnguyen.2014, akshatkumar, hclau}@smu.edu.sg, [‡] sheldon@cs.umass.edu

Abstract

Collective graphical models (CGMs) provide a framework for reasoning about a population of independent and identically distributed individuals when only *noisy* and *aggregate* observations are given. Previous approaches for inference in CGMs work on a junction-tree representation, thereby highly limiting their scalability. To remedy this, we show how the Bethe entropy approximation naturally arises for the inference problem in CGMs. We reformulate the resulting optimization problem as a difference-of-convex functions program that can capture different types of CGM noise models. Using the concave-convex procedure, we then develop a scalable message-passing algorithm. Empirically, we show our approach is highly scalable and accurate for large graphs, more than an order-of-magnitude faster than a generic optimization solver, and is guaranteed to converge unlike the previous message-passing approach NLBP that fails in several loopy graphs.

1 Introduction

Collective graphical models (CGMs) are a recently introduced formalism for inference and learning about a population of independent and identically distributed individuals when only *noisy* and *aggregate* observations are given (Sheldon and Dietterich, 2011). In many settings, such as in ecology, social sciences and transportation, data about each individual is rarely available due to privacy concerns or the difficulty of tracking each individual over time. As an example, the eBird database[1] contains observations about the count of birds at different locations and time across the North American region (Sheldon et al., 2007). The data released by Census Bureau may contain count-based aggregate information for privacy reasons. Similarly, the traffic data typically contains noisy aggregate count of vehicles at different locations (Morimura et al., 2013; Kumar et al., 2013). In such scenarios, CGMs can be used to model the individual-level behavior by doing inference and learning based on the available *noisy* and *aggregate* data.

A number of approaches have been proposed for inference in CGMs (Sheldon and Dietterich, 2011; Sheldon et al., 2013; Liu et al., 2014; Sun et al., 2015). Sheldon et al. develop a continuous convex relaxation of the MAP inference problem formulated over the junction tree derived from the individual model, and solve it using a generic optimization solver. Liu et al. develop a Gaussian approximation for CGMs and use Expectation-Propagation for inference. Sun et al. generalize the well known belief propagation algorithm (Pearl, 1988) to *nonlinear* belief propagation (NLBP) for CGMs. All such previous approaches require a junction tree representation for their correctness and convergence analysis. However, for large graphs, the tree-width can be high, which makes previous approaches computationally intractable.

In our work, we address this key shortcoming of previous approaches. We address the MAP inference for CGMs, which is also used as a sub-step for parameters learning (Sheldon et al., 2013) within the EM framework (Dempster et al., 1977). We show how the Bethe entropy approximation naturally arises for the MAP inference in CGMs similar to standard graphical models (Yedidia et al., 2005). A key difference for CGMs from standard graphical models is the presence of noise terms (i.e., the observed counts can be corrupted by some noise). The resulting optimization problem for CGM MAP inference is non-convex.

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

[1] `http://ebird.org/`

We show how this problem can be reformulated as a difference-of-convex functions (DC) program. As the DC representation of a given problem is not unique, we use additional variables and constraints such that the resulting DC program can address different types of noise models commonly used in the CGM framework (such as the Gaussian or Poisson noise). We use the concave-convex procedure (CCCP) (Yuille and Rangarajan, 2001; Yuille, 2002) to solve the resulting DC program using message-passing. The CCCP message-passing is over the individual model and does not require a junction tree. Thereby, it is highly scalable, and also guaranteed to converge.

Empirically, we test our approach on several (cyclic) graphs including a more realistic bird migration model (Sun et al., 2015) that takes into account the fact that destinations of different migratory birds can be different, and random and Ising grid graphs. We compare CCCP against a generic optimization solver to solve the DC program and the message passing approach NLBP (Sun et al., 2015). Even though NLBP is not guaranteed to converge on cyclic graphs, it is still well defined (like the standard BP), and converges for smaller cyclic graphs. Our results show that CCCP is more than an order-of-magnitude faster than the generic solver, and provides significantly more accurate results than NLBP. While NLBP fails to converge in several instances, CCCP is guaranteed to converge exhibiting better theoretical properties.

2 Collective Graphical Models

Collective graphical models (CGMs) describe the distribution of the aggregate statistics of a population of individuals sampled from a discrete graphical model (also known as the *individual* model). Let $G=(V,E)$ denote a pairwise Markov random field describing the individual model. Let $\boldsymbol{X}=(X_1,\ldots,X_{|V|})$ denote the random variables associated with each node in G. The joint-probability is:

$$p(\boldsymbol{x};\boldsymbol{\theta})=\Pr(\boldsymbol{X}=\boldsymbol{x};\boldsymbol{\theta})=\frac{1}{Z(\boldsymbol{\theta})}\prod_{(i,j)\in E}\phi_{ij}(x_i,x_j;\boldsymbol{\theta}) \quad (1)$$

where $\phi_{ij}(\cdot,\cdot;\boldsymbol{\theta})$ is the potential function for the edge (i,j) in G defined as per the parameters $\boldsymbol{\theta}$; $Z(\boldsymbol{\theta})$ denotes the partition function. Let the domain of each variable be denoted using $\mathcal{X}$.

Consider i.i.d. samples $\{\mathbf{x}^1,\ldots,\mathbf{x}^M\}$ drawn from the model G representing a population of M individuals. We can define the counts or contingency tables for this population as follows. Let $\mathbf{n}_i=(n_i(x_i):x_i\in\mathcal{X})$ represent the node counts, and $\mathbf{n}_{ij}=\big(n_{ij}(x_i,x_j):x_i,x_j\in\mathcal{X}\big)$ represent the edge counts for different edges. The counts $n_i(x_i)$ and $n_{ij}(x_i,x_j)$ are defined as:

$$n_i(x_i)=\sum_{m=1}^{M}\mathbb{I}\big(X_i^m=x_i\big) \quad (2)$$

$$n_{ij}(x_i,x_j)=\sum_{m=1}^{M}\mathbb{I}\big(X_i^m=x_i,X_j^m=x_j\big) \quad (3)$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. In CGMs, noisy observations $y_i(x_i),y_{ij}(x_i,x_j)$ about some subset of counts $\mathbf{n}$ can be provided. The probability $p\big(y(\cdot)|n(\cdot)\big)$ is referred to as the *noise model* for the CGMs. The typical noise model used for CGMs include the Poisson and the Gaussian noise. We assume that $p(\boldsymbol{y}|\mathbf{n})$ is log-concave in $\mathbf{n}$, which makes the negative log-likelihood convex in $\mathbf{n}$.

CGM Distribution We first describe the structure of the CGM distribution $p(\mathbf{n};\boldsymbol{\theta})$. The CGM distribution is defined over the junction tree $\mathcal{T}$ corresponding to the graph G. Each node t of this tree is associated with a clique $C_t\subseteq V$. Let $\mathcal{C}$ denote the set of all the cliques for the tree $\mathcal{T}$. If C and C' denote two adjacent cliques in $\mathcal{T}$, then $S=C\cap C'$ denotes a *separator*. Let $\mathcal{S}$ denote the set of all the separators for this junction tree. For any subset $C\subseteq V$, and a particular assignment $x_C\in\mathcal{X}^{|C|}$, we can define the counts $n_C(x_C)$ analogous to the node and edge counts as:

$$n_C(x_C)=\sum_{m=1}^{M}\mathbb{I}\big(X_C^m=x_C\big) \quad (4)$$

Using counts $n_C(x_C)$, we can define the contingency table $\mathbf{n}_C$ similar to tables $\mathbf{n}_i$, $\mathbf{n}_{ij}$ for nodes and edges of G. Let $\mathbf{n}=\{\mathbf{n}_A:A\in\mathcal{C}\cup\mathcal{S}\}$ denote the combined vector of clique and separator counts. The vector $\mathbf{n}$ is sufficient statistic of the population (Liu et al., 2014). The distribution over this vector is denoted as the *CGM distribution*.

As shown in (Liu et al., 2014), the CGM distribution is given as $p(\mathbf{n};\boldsymbol{\theta})=f(\mathbf{n};\boldsymbol{\theta})g(\mathbf{n})$ where we have:

$$f(\mathbf{n};\boldsymbol{\theta})=\frac{1}{Z(\boldsymbol{\theta})^M}\prod_{(i,j)\in E}\prod_{x_i,x_j}\phi_{ij}(x_i,x_j;\boldsymbol{\theta})^{n_{ij}(x_i,x_j)} \quad (5)$$

$$g(\mathbf{n})=M!\frac{\prod_{S\in\mathcal{S}}\prod_{x_S\in\mathcal{X}^{|S|}}\big(\mathbf{n}_S(x_S)!\big)^{\nu(S)}}{\prod_{C\in\mathcal{C}}\prod_{x_C\in\mathcal{X}^{|C|}}\mathbf{n}_C(x_C)!} \quad (6)$$

where $\nu(S)$ denotes the number of times S appears as a separator or the number of junction tree edges (t,t') for which $S=C_t\cap C_{t'}$. The distribution $p(\mathbf{n};\boldsymbol{\theta})$ is defined over the following set of constraints:

$$\sum_{x_C\in\mathcal{X}^{|C|}}\mathbf{n}_C(x_C)=M\ \forall C\in\mathcal{C} \quad (7)$$

$$\mathbf{n}_S(x_S)=\sum_{x_{C\setminus S}}\mathbf{n}_C(x_S,x_{C\setminus S})\ \forall x_S;\forall S\sim C\in\mathcal{T} \quad (8)$$

where $S \sim C \in \mathcal{T}$ implies that S is adjacent to C in the junction tree $\mathcal{T}$. We also have the constraint that $\mathbf{n}$ must be integer valued. Notice that the above two sets of constraints are similar to the constraints defining the marginal polytope for a graphical model (Wainwright and Jordan, 2008). The only difference being that the counts must sum to M instead of 1, and counts must be integers.

3 Bethe Approximation for CGMs

We now show how the Bethe entropy approximation (Yedidia et al., 2005) can be used for MAP inference in cyclic CGMs. In the MAP inference, we are given noisy observations $\boldsymbol{y}$ about some subset of sufficient statistic $\mathbf{n}$. Our goal in this work is to find the best count vector $\mathbf{n}$ that maximizes $p(\mathbf{n}|\boldsymbol{y};\boldsymbol{\theta}) \propto p(\mathbf{n};\boldsymbol{\theta})p(\boldsymbol{y}|\mathbf{n})$. That is:

$$\mathbf{n}^\star = \arg\max_{\mathbf{n}} \big[\log p(\mathbf{n};\boldsymbol{\theta}) + \log p(\boldsymbol{y}|\mathbf{n})\big] \tag{9}$$

To solve the above optimization problem, we first analyze the structure of $\log p(\mathbf{n};\boldsymbol{\theta}) = \log f(\mathbf{n};\boldsymbol{\theta}) + \log g(\mathbf{n})$, where $p(\cdot)$ is the CGM distribution. Using definitions (5), (6), we have:

$$\log f(\mathbf{n};\boldsymbol{\theta}) \propto \sum_{(i,j)} \sum_{x_i,x_j} \mathbf{n}_{ij}(x_i,x_j) \log \phi_{ij}(x_i,x_j) \tag{10}$$

where we have ignored terms that are independent of $\mathbf{n}$ such as $M \log Z(\boldsymbol{\theta})$. We further have:

$$\log g(\mathbf{n}) \propto \sum_{S\in\mathcal{S}} \sum_{x_S} \nu(S) \log\big(n_S(x_S)!\big) - \sum_{C\in\mathcal{C}} \sum_{x_C} \log\big(n_C(x_C)!\big)$$

As addressing integer counts $\mathbf{n}$ is challenging within an optimization framework directly, we make an approximation by making counts $\mathbf{n}$ continuous, as also used previously (Sheldon et al., 2013). For continuous $\mathbf{n}$, we can further use the Stirling's approximation as $\log n! \approx n \ln n - n$. Using these approximations, we can simplify $\log g(\mathbf{n})$ further as:

$$\begin{aligned}
&\log g(\mathbf{n}) \propto \sum_{S\in\mathcal{S}} \sum_{x_S} \nu(S)\big[n_S(x_S)\log n_S(x_S) - n_S(x_S)\big] - \\
&\sum_{C\in\mathcal{C}} \sum_{x_C} \big[n_C(x_C)\log n_C(x_C) - n_C(x_C)\big] \\
&= \sum_{S\in\mathcal{S}} \sum_{x_S} \nu(S) n_S(x_S)\log n_S(x_S) - \sum_{S\in\mathcal{S}} \sum_{x_S} \nu(S) n_S(x_S) \\
&- \sum_{C\in\mathcal{C}} \sum_{x_C} n_C(x_C)\log n_C(x_C) + \sum_{C\in\mathcal{C}} \sum_{x_C} n_C(x_C)
\end{aligned}$$

We can simplify the above expression by observing that as per constraints (7) we have $\sum_{x_C} n_C(x_C) = M$, and from constraints (8), we have $\sum_S n_S(x_S) = \sum_{x_C} n_C(x_C) = M$. Thus, we have the final simplified expression for $\log g(\mathbf{n})$ after ignoring terms independent of $\mathbf{n}$ as below:

$$\sum_{S\in\mathcal{S}} \sum_{x_S} \nu(S) n_S(x_S)\log n_S(x_S) - \sum_{C\in\mathcal{C}} \sum_{x_C} n_C(x_C)\log n_C(x_C)$$

Notice that the above expression subject to the constraints (7) and (8) is analogous to the entropy of a graphical model, the only difference being that counts sum up to M, rather than 1. We can now use the Bethe entropy to approximate this term, which is nicely decomposable along the nodes and edges of the individual model G. We have the following approximation:

$$\begin{aligned}
\log g(\mathbf{n}) \lessapprox &\sum_{i\in V} \sum_{x_i\in\mathcal{X}} \big(\nu(i)-1\big) n_i(x_i)\log n_i(x_i) - \\
&\sum_{(i,j)\in E} \sum_{x_i,x_j} n_{ij}(x_i,x_j)\log n_{ij}(x_i,x_j)
\end{aligned} \tag{11}$$

where $\nu(i)$ denotes the degree of the node i in the graph G. The above approximation represents a significantly more tractable form of $\log g(\mathbf{n})$ as all the terms are defined over the pairwise graph G, rather than the junction tree. Finally combining (10) and (11), we have:

$$\begin{aligned}
\log p(\mathbf{n};\boldsymbol{\theta}) \lessapprox &\sum_{(i,j)\in E} \sum_{x_i,x_j} \mathbf{n}_{ij}(x_i,x_j)\log\phi_{ij}(x_i,x_j) + \\
&\sum_{(i,j)\in E} H_{ij} + \sum_{i\in V} (1-\nu(i)) H_i
\end{aligned} \tag{12}$$

where $H_{ij} = -\sum_{x_i,x_j} n_{ij}(x_i,x_j)\log n_{ij}(x_i,x_j)$ and $H_i = -\sum_{x_i} n_i(x_i)\log n_i(x_i)$. The constraint set that each valid $\mathbf{n}$ must satisfy is given as:

$$n_i(x_i) = \sum_{x_j} n_{ij}(x_i,x_j)\ \forall j \in \mathrm{Nb}_i, \forall x_i, \forall i \in V \tag{13}$$

$$\sum_{x_i} n_i(x_i) = M, \forall i \in V \tag{14}$$

The above set of constraints for CGMs are similar to the constraints that define the local polytope for graphical models (Wainwright and Jordan, 2008).

4 Approximate MAP Inference

For the approximate CGM MAP inference, our goal is to solve the optimization problem:

$$\max_{\mathbf{n}} \big[\log \tilde{p}(\mathbf{n};\boldsymbol{\theta}) + \log p(\boldsymbol{y}|\mathbf{n})\big] \tag{15}$$

subject to constraints (13) and (14), and $\log \tilde{p}(\mathbf{n})$ is the approximate CGM distribution in (12). Addressing the expression $\log p(\boldsymbol{y}|\mathbf{n})$ is relatively easier as the assumption in CGMs is that different counts are corrupted independently by a univariate noise model $p(y|n)$. We assume that the noise model is log-concave in n, which is satisfied for different noise models such as Poisson and Gaussian use previously (Sheldon et al.,

2013; Liu et al., 2014). For example, the noise model for corrupted node contingency tables is given as:

$$p_{node}(\boldsymbol{y}|\mathbf{n}) = \prod_i \prod_{x_i} p\big(y_i(x_i)|n_i(x_i)\big)$$
$$\log p_{node}(\boldsymbol{y}|\mathbf{n}) = \sum_{i,x_i} \log p\big(y_i(x_i)|n_i(x_i)\big) = \sum_{i,x_i} l_i(n_i(x_i)) \tag{16}$$

where we have used $l_i(n_i(x_i))$ to denote the log of the noise model without being tied to any specific model. The noise model for edge counts $n_{ij}(\cdot,\cdot)$ can be defined analogously. We now write the approximate MAP objective function (15) using expressions (12) and (16)

$$\min_{\{\mathbf{n}_i,\mathbf{n}_{ij},\mathbf{z}_i\}} -\sum_{(i,j)\in E}\sum_{x_i,x_j} n_{ij}(x_i,x_j)\log\phi_{ij}(x_i,x_j) - \sum_{(i,j)\in E} H_{ij}$$
$$-\sum_{i\in V}(1-\nu(i))H_i - \sum_{i\in V}\sum_{x_i\in\mathcal{X}} l_i(z_i(x_i)) \tag{17}$$
$$\text{Subject to: constraints (13) and (14)} \tag{18}$$
$$z_i(x_i) = n_i(x_i)\ \forall x_i, \forall i \in V \tag{19}$$

where we have included additional (redundant) variables $z_i(\cdot)$ for node counts, and included the constraint $z_i(\cdot) = n_i(\cdot)$ to make them equal to $\mathbf{n}_i$. These redundant variables and constraints would help later when we derive the CCCP algorithm to solve the above program, and make sure that resulting updates can incorporate different types of CGM noise models. For ease of exposition, we show our derivations for the noise added to node contingency tables; edge noise can be addressed analogously by creating such additional $z_{ij}(\cdot,\cdot)$ variables.

4.1 DC Programming for MAP Inference

The problem (17) can be solved using a generic nonlinear optimization solver. However, as we show empirically, such a strategy is not scalable to large graphs. Therefore, our goal is solve the problem (17) using message-passing. To achieve this, we first reinterpret (17) as a difference of convex functions (DC) program. Consider the optimization problem: $\min\{h(x) : x \in \Omega\}$, where $h(x) = u(x) - v(x)$ is an arbitrary function with u, v being real-valued, differentiable *convex* functions and Ω being a convex constraint set. Such a program is called a *DC program*. We decompose (17) as a difference of two convex functions u and v below:

$$u(\mathbf{n},\mathbf{z}) = -\sum_{(i,j)\in E}\sum_{x_i,x_j} n_{ij}(x_i,x_j)\log\phi_{ij}(x_i,x_j) - \sum_{(i,j)\in E} H_{ij}$$
$$-\sum_{i\in V} H_i - \sum_{i\in V}\sum_{x_i\in\mathcal{X}} l_i(z_i(x_i)) \tag{20}$$
$$v(\mathbf{n},\mathbf{z}) = -\sum_{i\in V}\nu(i)H_i \tag{21}$$

We next show how the CCCP framework can be used to solve the above DC program using message-passing.

4.2 CCCP For MAP Inference

The CCCP method provides an iterative procedure to solve the DC problem $\min\{h(x) = u(x) - v(x) : x \in \Omega\}$ by generating a sequence of points x^k by solving the following *convex* program:

$$x^{k+1} = \arg\min_x \{u(x) - x^T\nabla v(x^k) : x \in \Omega\} \tag{22}$$

Each iteration of CCCP decreases the objective function $h(x)$ and converges to a stationary point (Sriperumbudur and Lanckriet, 2009). The optimization problem that CCCP solves iteratively for the CGM MAP inference is:

$$\min_{\mathbf{n},\mathbf{z}} -\sum_{(i,j)\in E}\sum_{x_i,x_j} n_{ij}(x_i,x_j)\log\phi_{ij}(x_i,x_j) - \sum_{(i,j)\in E} H_{ij} - \sum_{i\in V} H_i$$
$$-\sum_{i\in V}\sum_{x_i\in\mathcal{X}} l_i(z_i(x_i)) - \sum_{i\in V}\sum_{x_i} n_i(x_i)\nabla_{n_i(x_i)}v \tag{23}$$

subject to constraints (13), (14) and (19); $\nabla_{n_i(x_i)}v$ represents the gradient of the function $v(\mathbf{n},\mathbf{z})$ w.r.t. $n_i(x_i)$ from previous iteration's solution. Notice that (23) is a convex optimization problem, and constraints are linear resulting in zero duality gap. Therefore, we solve the Lagrangian dual of this problem, which has a simpler structure amenable for solving by message-passing.

We start by writing the Lagrangian function of the problem (23). We use dual variables $\lambda_{ij}(x_i)$ for marginalization constraints (13), λ_i for normalization constraints (14), and $\lambda_i(x_i)$ for the equality constraints (19). The Lagrangian function is given as:

$$L(\mathbf{n},\mathbf{z},\lambda) = -\sum_{(i,j)\in E}\sum_{x_i,x_j} n_{ij}(x_i,x_j)\log\phi_{ij}(x_i,x_j) - \sum_{(i,j)\in E} H_{ij}$$
$$-\sum_{i\in V} H_i - \sum_{i\in V}\sum_{x_i\in\mathcal{X}} l_i(z_i(x_i)) - \sum_{i\in V}\sum_{x_i} n_i(x_i)\nabla_{n_i(x_i)}v$$
$$-\sum_{i\in V}\sum_{j\in \mathrm{Nb}_i}\sum_{x_i\in\mathcal{X}} \lambda_{ij}(x_i)\Big[n_i(x_i) - \sum_{x_j} n_{ij}(x_i,x_j)\Big]$$
$$-\sum_{i\in V}\lambda_i\Big[\sum_{x_i} n_i(x_i) - M\Big] - \sum_{i\in V}\sum_{x_i}\lambda_i(x_i)\Big[z_i(x_i) - n_i(x_i)\Big]$$

The next step is compute the dual function $q(\lambda) = \min_{\mathbf{n},\mathbf{z}} L(\mathbf{n},\mathbf{z},\lambda)$. This is another optimization problem. However, it is an easier unconstrained optimization problem, and can be solved by setting partial derivatives of $L(\cdot)$ to zero w.r.t. different variables. Solving for $\mathbf{n}$, we get:

$$n_{ij}(x_i,x_j) = \phi_{i,j}(x_i,x_j)e^{-\lambda_{ij}(x_i)-\lambda_{ji}(x_j)-1} \tag{24}$$
$$n_i(x_i) = e^{\lambda_i + \sum_{j\in\mathrm{Nb}_i}\lambda_{ij}(x_i) + \nabla_{n_i(x_i)}v - \lambda_i(x_i) - 1} \tag{25}$$

Notice that the above two equations are independent of the noise model used for the CGM. Such a decoupling was possible by the introduction of $z(\cdot)$ variables.

Algorithm 1: CGM MAP Inference

1 **Algorithm** `CCCP Message Passing()`
2 **Initialize:** $\lambda_{ij}(x_i) \leftarrow 0 \; \forall (i,j) \in E$
$\nabla_{n_i(x_i)} v = 0, \lambda_i, \lambda_i(x_i) \leftarrow 0 \; \forall i \in V, x_i \in \mathcal{X}$
3 **repeat**
4 **repeat**
5 **for** *each edge* $(i,j) \in E$ **do**
6 `UpdateEdgeVars(`*i, j*`)`
7 `UpdateEdgeVars(`*j, i*`)`
8 **for** *each node* $i \in V$ **do**
9 `UpdateNodeVars(`*i*`)`
10 **until** ***inner loop converges***
11 $n_i(x_i) \leftarrow e^{\lambda_i + \sum_{j \in \text{Nb}_i} \lambda_{ij}(x_i) + \nabla_{n_i(x_i)} v - \lambda_i(x_i) - 1}$
12 $n_{i,j}(x_i, x_j) \leftarrow \phi_{i,j}(x_i, x_j) e^{-\lambda_{ij}(x_i) - \lambda_{ji}(x_j) - 1}$
13 $\nabla_{n_i(x_i)} v \leftarrow \nu(i)\big[1 + \log n_i(x_i)\big] \; \forall x_i, \forall i \in V$
14 **until** ***outer loop converges***
15 **return** Count table $\mathbf{n}$
16 **Procedure** `UpdateEdgeVars(`*i, j*`)`
17 $\lambda_{ij}(x_i) \leftarrow \frac{1}{2}[\log\big(\sum_{x_j} \phi_{i,j}(x_i, x_j) e^{-\lambda_{ji}(x_j)}\big) - \lambda_i - \sum_{k \in Nb_i \setminus \{j\}} \lambda_{ik}(x_i) - \nabla_{n_i(x_i)} v + \lambda_i(x_i)] \; \forall x_i \in \mathcal{X}$
18
19 **Procedure** `UpdateNodeVars(`*i*`)`
20 $\lambda_i \leftarrow \log M - \log \sum_{x_i} e^{\sum_{j \in \text{Nb}_i} \lambda_{ij}(x_i) + \nabla_{n_i(x_i)} v - \lambda_i(x_i) - 1}$
21 Poisson noise model: $\lambda_i(x_i) \leftarrow \alpha_i(x_i) - W\Big(y_i(x_i) e^{-\lambda_i - \sum_{j \in \text{Nb}_i} \lambda_{ij}(x_i) - \nabla_{n_i(x_i)} v + 1 + \alpha_i(x_i)}\Big) \; \forall x_i$
22 Gaussian noise model: $\lambda_i(x_i) \leftarrow -\frac{y_i(x_i)}{\sigma_i^2(x_i)} +$
23 $W\Big(\frac{1}{\sigma_i^2(x_i)} e^{\lambda_i + \sum_{j \in \text{Nb}_i} \lambda_{ij}(x_i) + \nabla_{n_i(x_i)} v - 1 + \frac{y_i(x_i)}{\sigma_i^2(x_i)}}\Big) \; \forall x_i$

When we set partial derivative of L w.r.t. $z_i(x_i)$ to zero, we get the following condition:

$$-\nabla_{z_i(x_i)} l_i(z_i(x_i)) - \lambda_i(x_i) = 0 \tag{26}$$

For solving the above equation, we need to know the structure of the noise model. We show below how to solve it for two commonly used noise models—Poisson and Gaussian.

Poisson Noise The Poisson noise in the count of $z_i(x_i)$ with detection rate $\alpha_i(x_i)$ is given as:

$$p(y_i(x_i)|z_i(x_i)) = \frac{\alpha_i(x_i) z_i(x_i)^{y_i(x_i)} e^{-\alpha_i(x_i) z_i(x_i)}}{y_i(x_i)!}$$

Substituting the value of $\nabla \log p(y_i(x_i)|z_i(x_i))$ to (26), we get the value of $z_i(x_i)$ as follows:

$$z_i(x_i) = \frac{y_i(x_i)}{\alpha_i(x_i) - \lambda_i(x_i)} \tag{27}$$

Gaussian Noise The Gaussian noise in the count of $z_i(x_i)$ with mean $z_i(x_i)$ and variance $\sigma_i^2(x_i)$ is:

$$p(y_i(x_i)|z_i(x_i)) = \frac{1}{\sigma_i(x_i)\sqrt{2\pi}} e^{-\frac{(y_i(x_i) - z_i(x_i))^2}{2\sigma_i^2(x_i)}}$$

Substituting the value of $\nabla \log p(y_i(x_i)|z_i(x_i))$ to (26), we get the value of $z_i(x_i)$ as follows:

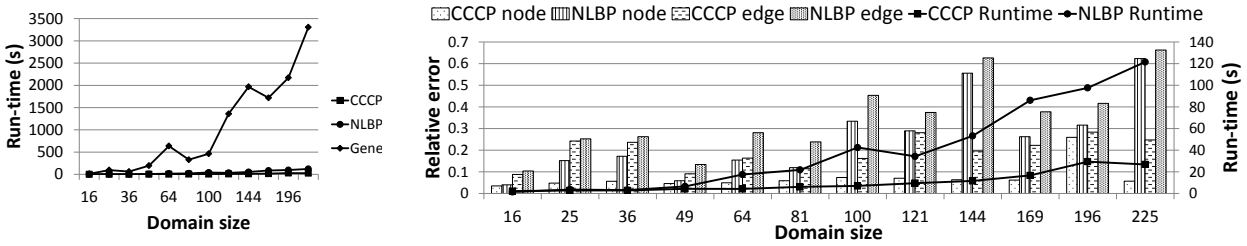
$$z_i(x_i) = y_i(x_i) + \sigma_i^2(x_i)\lambda_i(x_i) \tag{28}$$

In a similar way as above, we can derive the updates for any other log-concave and differentiable CGM noise function.

Optimizing the Dual Function Once we substitute the values of $n_i(\cdot)$, $n_{ij}(\cdot,\cdot)$ and $z_i(\cdot)$ from (24), (25) and (27) or (28) to the Lagrangian function L, we get the dual function $q(\lambda)$. The dual optimization problem is $\max_\lambda q(\lambda)$. We can solve this iteratively by using the block-coordinate ascent strategy (Bertsekas, 1999), wherein we fix all the dual variables except one, and then optimize the function $q(\lambda)$ w.r.t. the one dual variable. Such a strategy is guaranteed to converge to the optimal primal solution as the dual is differentiable and concave for each λ variable, and there is a unique solution for each block-coordinate ascent step (Bertsekas, 1999). We show the final updates for different dual variables in algorithm 1.

The CCCP message passing for CGMs takes the form of a double loop algorithm (as shown in alg. 1). The inner loop runs from line 4 till line 10. The inner loop performs the block coordinate ascent step for each node and edge dual variable for a fixed gradient ∇v from the previous iteration's parameters $\mathbf{n}$. The update equation for edge dual variables (that enforce the marginalization constraints (13)) $\lambda_{ij}(\cdot)$ is provided in the function `UpdateEdgeVars`$(\cdot)$. The function `UpdateNodeVars`$(\cdot)$ updates the dual variables λ_i that enforce the normalization constraints (14). Notice that both these updates do not depend on the CGM noise model. Finally, the update equations for $\lambda_i(x_i)$ that enforce the equality constraints (19) are provided in function `UpdateNodeVars` for Gaussian and Poisson models. The function $W(\cdot)$ is the Lambert's W function (or the productLog function) (Corless et al., 1996), which is used to solve the equation $xe^x = k$, where k is some positive constant. The solution for such a equation is given as $x = W(k)$.

The inner loop convergence of CCCP is detected when the constraint violation falls below a certain threshold. The convergence of the outer loop is detected when the quality improvement over iterations is below a threshold. The runtime and the space complexity of each CCCP inner loop increases linearly with the number of edges in the graph. Empirically, we show that inner loop converges within 10 iterations even for large graphs. Thus, our approach is highly scalable.



(a) Runtime for generic solver

(b) Error and Run-time of CCCP and NLBP

Figure 1: Bird Migration results

5 Experiments

We evaluate our CCCP based message passing on three sets of instances (1) bird migration benchmarks from (Liu et al., 2014; Sun et al., 2015), (2) random grid graphs of varying sizes and (3) Ising grid graphs. We compare CCCP against the Matlab's generic optimization solver to directly solve the DC program and the message passing approach NLBP from (Sun et al., 2015). The NLBP algorithm, like the standard BP, may not converge for cyclic graphs, but it is still well defined. Sun et al. (2015) show that when NLBP converges, it converges to a local optimum of the problem (17). The NLBP is also a double loop algorithm where each inner loop runs the standard BP, and the outer loop computes the gradient of the nonlinear free energy of CGMs (Sun et al., 2015).

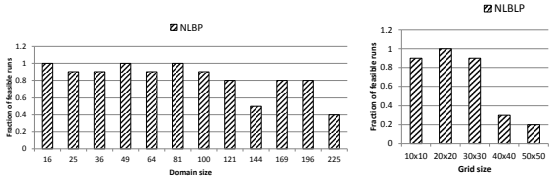
Observation Model: In our experiments, we use Gibb's sampling procedure to generate values of individuals in the population from cyclic graphical model. Only noisy observations for node counts are given as input to solver, edge counts are hidden. Given the noisy observation of node counts, we consider the inference task to infer underlying node and edge counts.

We focus in the experiments on the following aspects of the CCCP approach. First, we empirically show that CCCP is highly scalable for large graphs. The Matlab's generic solver's runtime increases significantly with the problem size, and is more than an order-of-magnitude slower than CCCP for larger bird migration problems. CCCP is also faster than NLBP on several grid graphs as NLBP takes many iterations to find a feasible solution, whereas CCCP's inner loop converges much faster leading to speedups over NLBP. Second, we show that CCCP finds accurate results. The relative error, $||\mathbf{n}^{\text{CCCP}} - \mathbf{n}^{\text{True}}||_1 / ||\mathbf{n}^{\text{True}}||_1$, is about 10% for most instances. This shows that the Bethe entropy approximation and the CCCP message passing are effective even in the presence of high number of cycles in the graph.

Bird Migration Benchmark We evaluate the speed and the accuracy of CCCP on bird migration benchmarks generated in a similar manner as (Liu et al., 2014; Sun et al., 2015). The previous work generates data from a T-step chain-structured CGM to simulate the migration of a population of $M = 1000$ birds from the bottom-left corner to the top-right corner of an $L \times L$ grid map for a total of T time steps. Let x_t denote the location of a bird at time t in the $L \times L$ map. The transition probability between cells x_t and x_{t+1} comes from a logistic regression formula that has several covariates such as the distance among locations, consistency of the wind direction with transition movement among others. We adopt such features and their weights from the implementation provided by Sun et al.

We make such a chain-structured CGM more realistic by removing the implicit assumption in previous models that all the birds migrate to the same destination. This results in a *cyclic* CGM, rather than a chain structure. We assume that each bird can choose any destination within a specified group of cells in the top-right map region with some probability. To simulate this dependence on different destinations, we create an edge between every node $x_t \forall t = 1$ to $T-1$ to x_T. We set the potential $\phi_{t,T}(x_t, x_T) \propto \frac{1}{d(x_t, x_T)} U(x_T)$, where $d(x_t, x_T)$ denotes the distance between locations x_t and x_T, and $U(x_T) \in [0, 1]$ is a unary function that specifies the relative preference among different destinations x_T. For experiments, we assume that $U(x_T)$ follows a uniform distribution.

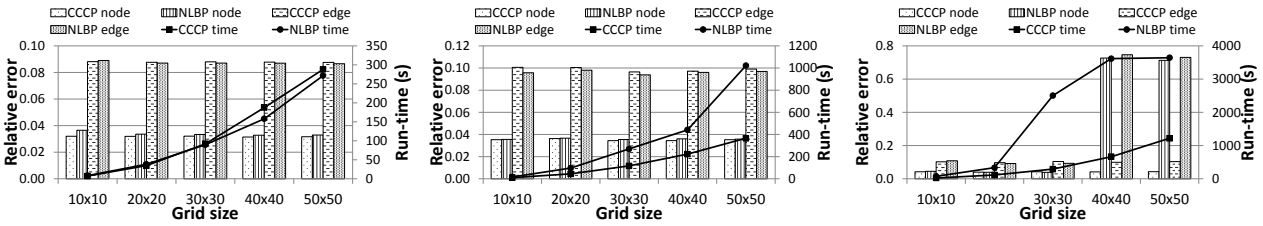
Figure 1(b) shows the accuracy of CCCP and NLBP for different map sizes ('16'=4×4 to '225'=15×15) and horizon $T = 15$ on the primary y-axis. We used



(a) Bird migration domain

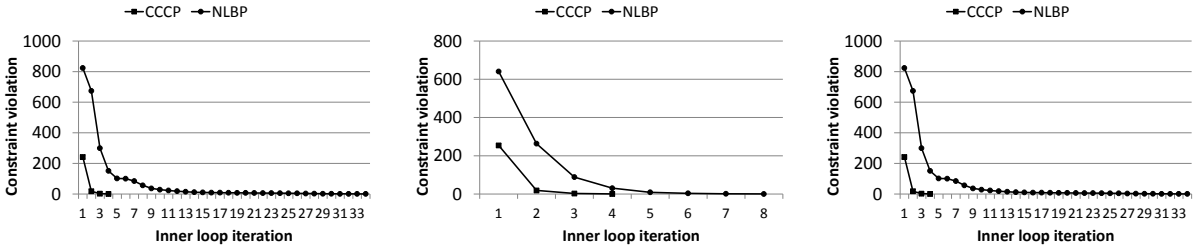
(b) Random grid

Figure 2: Fraction of total NLBP runs that resulted in a feasible solution for the bird migration domain, and random grids with $\beta = 3$



(a) $\beta = 1$ (b) $\beta = 2$ (c) $\beta = 3$

Figure 3: Quality and runtime comparisons for random grid graphs



(a) $\beta = 1$ (b) $\beta = 2$ (c) $\beta = 3$

Figure 4: Average constraint violation for CCCP and NLBP inner loops for random grid graphs

Gibb's sampling to sample a population of $M = 1000$ birds. After generating the node and edge count tables, we generated noisy observations for node counts $y_i \sim \text{Pois}(\alpha n_i)$ with detection rate $\alpha = 1$.

The accuracy of Matlab's solver was very close to that of CCCP for each instance and thus, is not plotted. The x-axis shows the map size and y-axis shows the relative error, $||\mathbf{n}^{\text{alg}} - \mathbf{n}^{\text{True}}||_1 / ||\mathbf{n}^{\text{True}}||_1$, between the true counts and the final solution of each algorithm. We show both the error in node counts ('node error') and edge counts ('edge error'). Each data point is an average of 10 runs. In each run, the destination preference $U(\cdot)$ was sampled randomly.

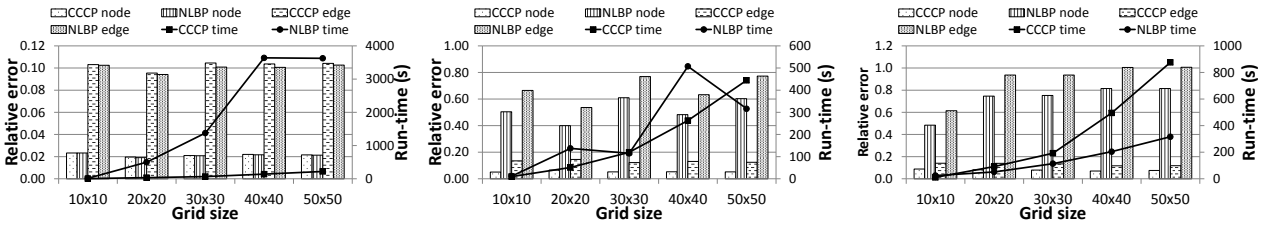
From figure 1(b), we can clearly see that for small map sizes (until '49' or 7×7 map), both CCCP and NLBP have similar accuracy. However, as the map size increases from 8×8 to 15×15, the accuracy of NLBP goes down significantly. For the map size 15×15, NLBP error was very high, more than 60% for both nodes and edges. In contrast, CCCP provides highly accurate results. The average relative node error across all the map sizes for CCCP is $\approx 6\%$ and the average relative edge error is $\approx 19\%$. The node error is smaller than the edge error as noisy observations are provided for nodes, whereas no observations are provided for the edges. These sets of results show that CCCP was highly accurate even for larger instances.

There were several bird migration instances for which NLBP failed to find a single feasible solution (that satisfied all the marginalization and normalization constraints on $\mathbf{n}$ up to a specified threshold). Figure 2(a) shows the failure rate of NLBP for different map sizes on the x-axis. The y-axis shows the total fraction of instances for which NLBP found at least one feasible solution. We can see that for larger maps (from '81' or 9×9 onwards), NLBP failed to find a feasible solution for more than 20% of instances. These results clearly show that for larger problems, CCCP provides a significantly better alternative than NLBP with much better convergence guarantees and stability.

Figure 1(a) shows the runtime comparisons between Matlab's generic solver and CCCP. Even though the accuracy of matlab's solver was similar to CCCP, the runtime for the generic solver increases significantly with the increasing problem size. CCCP is more than an order-of-magnitude faster than the generic solver, and is thus much more scalable.

Random Grid graphs We also compare NLBP and CCCP on random grid graphs. We generated grids of varying sizes from the smaller 10×10 grids to large 50×50 grids. The domain size of each variable is 4. The log-potential for each setting of edge variables was sampled uniformly from $[-\beta, \beta]$. We used three settings of $\beta \in \{1, 2, 3\}$. The population size was $M = 1000$, and the noise model settings were the same as for bird migration benchmarks. All data points are an average over 10 runs. Such grid graphs have significantly higher number of cycles than the bird migration instances, thereby rigorously testing the accuracy of Bethe approximation and the CCCP message passing.

Figures 3(a-c) show the accuracy of CCCP and NLBP for varying grid sizes and β values, and also the runtime on the secondary y-axis. For smaller values of $\beta \in \{1, 2\}$ in figures 3(a-b), both NLBP and CCCP



(a) $\beta = 1$ (b) $\beta = 2$ (c) $\beta = 3$

Figure 5: Quality and runtime comparisons for Ising grid graphs

were reasonably accurate. The relative edge error for CCCP and NLBP both for these settings was $< 10\%$. However, for $\beta = 3$, NLBP's performance degrades significantly, with relative error becoming quite high ($\approx 50\%$) for larger grid sizes (40×40 and 50×50). In contrast, CCCP had similar accurate performance as for the other settings of β. We noticed that with a higher value of β, the sparsity of the sampled data increases, which may have caused NLBP to become unstable for higher β values.

Figure 2(b) shows the failure rate for NLBP for different grid sizes for $\beta = 3$. We can see clearly that as the grid size increases, the failure rate of NLBP also increases significantly (more than 60% for 40×40 and 50×50 grids). Thus, for such random graphs also CCCP provided highly accurate results and proved to be more stable than NLBP for a range of problem sizes.

In terms of the runtime (secondary y-axis), figures 3(a-c) show that the runtime of NLBP increases significantly with the increasing value of β (e.g., for 50×50 grid in figure 3(c), NLBP is significantly slower than CCCP). Figures 4(a-c) provide further insight to this observation. In these figures, we show the average number of inner loops required per outer loop iteration for both CCCP and NLBP. The x-axis denotes the inner loop iteration number, and y-axis denotes the average constraint violation over all the instances for NLBP and CCCP. We can observe that for higher values of β, NLBP requires higher number of inner loops (per outer loop iteration) to reach a constraint violation threshold close to zero. Such higher number of inner loops lead to a large increase in the runtime for NLBP. In contrast, CCCP requires far fewer inner loops, typically less than 6, to reach a solution within the violation threshold.

Ising Grid graphs We also compared CCCP and NLBP on Ising grid graphs. For these graphs, each variables x_i can take two values $\{1, -1\}$. The log-potential for the variable setting (x_i, x_j) is set to $x_i x_j \beta_{ij}$, where each β_{ij} is sampled uniformly from the interval $[-\beta, \beta]$. The population size and the noise model is the same as for random graphs.

Figures 5(a-c) show the accuracy and the runtime comparisons between CCCP and NLBP for a range of grid sizes. For these instances too, the higher values of $\beta \in \{2, 3\}$ lead to instability in NLBP resulting in high relative error ($> 50\%$) as shown in Figures 5(b-c). The CCCP on the other hand proves to be accurate across a range of settings with the relative error about 10%. For Ising graphs, NLBP finds feasible solutions and converges for all the instances. However, the solution at convergence may not be accurate enough as shown in figures 5(b-c).

As per the runtime comparisons, NLBP can sometimes converge faster than CCCP (as shown in Figure 5(c), for grid size 40×40 and 50×50). However, the accuracy of NLBP is much worse than CCCP.

6 Conclusion

In this work, we addressed the problem of MAP inference in collective graphical models, which are a framework for reasoning with noisy aggregate data. Previous approaches for inference in CGMs work on a junction-tree representation, thereby highly limiting their scalability. We addressed this key bottleneck in our work by introducing the Bethe entropy approximation for the CGM MAP inference problem. We showed how to interpret the resulting optimization problem as a DC program, and derived the CCCP algorithm to solve it using message passing. Our extensive empirical evaluations on multiple benchmarks showed that CCCP was much faster than a generic optimization solver. The CCCP approach was also highly accurate as compared to the previous message passing approach NLBP across different benchmarks, and guaranteed to converge unlike NLBP which failed to converge on several instances. Thus, our work advances the applicability of CGMs to large graphs by providing an effective and accurate method for inference.

Acknowledgements

This research is funded by the National Research Foundation Singapore under its Corp Lab@University scheme; Daniel Sheldon is supported by the National Science Foundation under Grant No. 1125228.

References

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific, Cambridge, MA, USA.

Corless, R. M., Gonnet, G. H., Hare, D. G., Jeffrey, D. J., and Knuth, D. E. (1996). On the Lambert W function. In *Advances in Computational Mathematics*, pages 329–359.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical society, Series B*, 39(1):1–38.

Kumar, A., Sheldon, D., and Srivastava, B. (2013). Collective diffusion over networks: Models and inference. In *International Conference on Uncertainty in Artificial Intelligence*, pages 351–359.

Liu, L., Sheldon, D., and Dietterich, T. (2014). Gaussian approximation of collective graphical models. In *International Conference on Machine Learning*, pages 1602–1610.

Morimura, T., Osogami, T., and Idé, T. (2013). Solving inverse problem of markov chain with partial observations. In *Advances in Neural Information processing Systems*, pages 1655–1663.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers Inc.

Sheldon, D., Elmohamed, M. A. S., and Kozen, D. (2007). Collective inference on markov models for modeling bird migration. In *Advances in Neural Information Processing Systems*, pages 1321–1328.

Sheldon, D., Sun, T., Kumar, A., and Dietterich, T. G. (2013). Approximate inference in collective graphical models. In *International Conference on Machine Learning*, pages 1004–1012.

Sheldon, D. R. and Dietterich, T. G. (2011). Collective graphical models. In *Advances in Neural Information Processing Systems*, pages 1161–1169.

Sriperumbudur, B. and Lanckriet, G. (2009). On the convergence of the concave-convex procedure. In *Advances in Neural Information Processing Systems*, pages 1759–1767.

Sun, T., Sheldon, D., and Kumar, A. (2015). Message passing for collective graphical models. In *International Conference on Machine Learning*, pages 853–861.

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

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312.

Yuille, A. L. (2002). CCCP algorithms to minimize the bethe and kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722.

Yuille, A. L. and Rangarajan, A. (2001). The concave-convex procedure (CCCP). In *Advances in Neural Information Processing Systems*, pages 1033–1040.