C^4: Exploring Multiple Solutions in Graphical Models by Cluster Sampling

Jake Porway and Song-Chun Zhu

Abstract—This paper presents a novel Markov Chain Monte Carlo (MCMC) inference algorithm called C^4 – Clustering with Cooperative and Competitive Constraints for computing *multiple solutions* from posterior probabilities defined on graphical models, including Markov random fields (MRF), conditional random fields (CRF) and hierarchical models. The graphs may have both positive and negative edges for cooperative and competitive constraints. C^4 is a probabilistic clustering algorithm in the spirit of Swendsen-Wang [34]. By turning the positive edges on/off probabilistically, C^4 partitions the graph into a number of connected components (ccp's) and each ccp is a coupled sub-solution with nodes connected by positive edges. Then by turning the negative edges on/off probabilistically, C^4 obtains composite ccp's (called $cccp$'s) with competing ccp's connected by negative edges. At each step C^4 flips the labels of all nodes in a $cccp$ so that nodes in each ccp keep the same label while different ccp's are assigned different labels to observe both positive and negative constraints. Thus the algorithm can jump between multiple competing solutions (or modes of the posterior probability) in a single or a few steps. It computes multiple distinct solutions to preserve the intrinsic ambiguities and avoids premature commitments to a single solution that may not be valid given later context. C^4 achieves a mixing rate faster than existing MCMC methods, such as various Gibbs samplers [15], [26] and Swendsen-Wang cuts [2], [34]. It is also more "dynamic" than common optimization method such as ICM [3], LBP [21], [37], and graph cuts [4], [20]. We demonstrate the C^4 algorithm in line drawing interpretation, scene labeling, and object recognition.

1. Introduction

1.1. Motivations and Objective

Many vision tasks, such as scene labeling [22], [31], [32], object detection/recognition [11], [36], segmentation [8], [35], and graph matching [6], [24] are formulated as energy minimization (or maximum a posteriori probability) problems defined on graphical models – Markov random fields [3], [15], conditional random fields [22], [23], or hierarchical graphs [14], [40]. These optimization problems become exceedingly difficult when there are multiple solutions, i.e. distinct modes with high probabilities and, in some cases, equal probability.

Fig. 1 shows examples of typical scenarios that have multiple, equally likely solutions in the absence of further context. The top row shows the well-known Necker Cube which has two valid 3D interpretations. The middle row is the Wittgenstein illusion, in which the drawing can appear to be either a duck or a rabbit. Without further context, we cannot determine the correct labeling. The bottom row shows an aerial image for scene labeling. It can be explained as either a roof with vents or a parking lot containing cars.

Computing multiple solutions is important for preserving the intrinsic ambiguities and avoiding early commitment to a single solution which, even if it's currently the globally optimal one, may turn out to be less favorable when later context arrives. However, it is a persistent challenge to enable algorithms to climb out of local optima and to jump between solutions far apart in the state space. Popular energy minimization algorithms, such as Iterative Conditional Modes (ICM) [3], Loopy Belief Propagation (LBP) [21], [37], and graph cuts [4], [20] compute one solution and thus do not address this problem. Existing

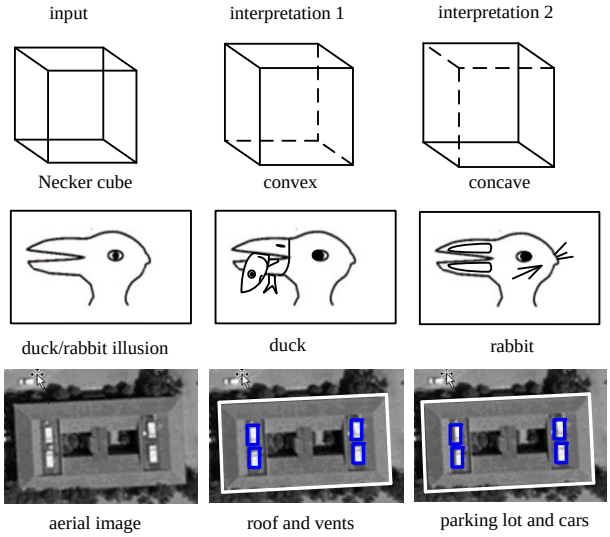


Fig. 1: Problems with multiple solutions: (top) the Necker Cube; (Middle) the Wittgenstein illusion; and (Bottom) An aerial image interpreted as either a roof with vents or a parking lot with cars. Ambiguities should be preserved until further context arrives.

• *Jake Porway was a Ph.D student with the Dept. of Statistics, UCLA when the paper was submitted and is now a data scientist with the R& D Division at New York Times, and Song-Chun Zhu is a professor with the Depts of Statistics and Computer Science, UCLA, and the Lotus Hill Research Institute*

MCMC algorithms, such as various Gibbs samplers [15], [26], DDMCMC [35], and Swendsen-Wang cuts [2], [34], promise global optimization and ergodicity in the state space, but often need long waiting time in moving between distinct modes, which needs a sequence of lucky moves up the energy landscape before it goes down.

In this paper, our objective is to develop an algorithm that can discover multiple solutions by jumping out of equal probability states and thus preserve the ambiguities on rather general settings:

1) The graph can be flat, such as a MRF or CRF, or hierarchical, such as a parse graph.
2) The graph may have positive (cooperative) and negative (competitive or conflicting) edges for both hard or soft constraints.
3) The probability (energy) defined on the graph is quite general, even with energy terms involving more than two nodes.

In vision, it is safe to assume that the graph is locally connected and we do not consider the worst case scenario where graphs are fully connected.

1.2. Related Work in the Literature

In the 1970s, many problems, including line drawing interpretation and scene labeling, were posed as constraint satisfaction problems (CSPs). The CSPs were either solved by heuristic search methods [30] or constraint propagation methods [1], [28]. The former keeps a list of open nodes for plausible alternatives and can backtrack to explore multiple solutions. However, the open list can become too long to maintain when the graph is large. The latter iteratively updates the labels of nodes based on their neighbors. One well-known constraint propagation algorithm is the relaxation labeling method by Rosenfeld, Hummel, and Zucker in 1976 [32].

In the 1980s, the famous Gibbs sampler – a probabilistic version of relaxation labeling – was presented by Geman and Geman in 1984 [15]. The update of labels is justified in a solid MCMC and MRF framework and thus is guaranteed to sample from the posterior probabilities. In special cases, the Gibbs sampler is equal to belief propagation [30] for polytrees and to dynamic programming in chains. The Gibbs sampler is found to slow down critically when a number of nodes in the graph are strongly coupled.

Fig. 2 illustrates an example of the difficulty with strongly coupled graphs using the Necker Cube. The six internal lines of the figure are divided into two coupling groups: (1-2-3) and (4-5-6). Lines in each group must have the same label (concave or convex) to be valid as they share the two 'Y'-junctions. Thus, updating the label of a single line in a coupled group does not move at all, unless we update the label of the whole group together, i.e. all six labels in one step.

The problem is that we don't know which nodes in the graph are coupled and to what extent they are coupled for general problems with large graphs. In 1987, a breakthrough came from two physicists, Swendsen and Wang [34], who proposed a cluster sampling technique. The Swendsen-Wang (SW) method finds coupled groups, called "clusters", dynamically by turning the edges in the graph on/off according to the probabilities defined on these edges. The edge probability measures the coupling strengths. Unfortunately, their algorithm only works for the Ising and Potts models. We will discuss the SW method in later sections.

There were numerous attempts made to improve MCMC methods in the 1990's (see Liu [25] for surveys), such as the block Gibbs sampler [26]. Green formulated reversible jumps in 1995 [17] following the jump-diffusion algorithm by Grenander and Miller in 1994 [18]. In 1999, Cooper and Frieze analyzed the convergence speed of SW using a path coupling technique and showed that the SW method has a polynomial mixing time when the nodes in the graph are connected to a constant number of neighbors [7]. Nevertheless, it was also shown that SW could mix slowly under conditions when graphs were heavily or fully connected [16].

In the 2000s, a few non-MCMC methods generated remarkable impacts on the vision community. For example, the loopy belief propagation (LBP) algorithm by Weiss et. al. [37] and the graph cut algorithms by Boykov, Kolmogorov, et. al.[4], [20]. These algorithms are very fast and work well on special class of graph structures and energy functions. In addition, techniques such as survey propagation [5] have had great success in statistical physics. In the case of multimodal energy functions, however, it can be difficult for these techniques to converge properly, as we will see.

On the MCMC side, Tu and Zhu developed the Data-Driven Markov Chain Monte Carlo (DDMCMC) algorithm for image segmentation in 2002 [35], which uses bottom-up discriminative probabilities to drive the Markov chain moves. They also developed a "K-adventurer" procedure to keep multiple solutions. The DDMCMC method was also used by Dellaert [29] for tracking bee dances. Dellaert also used MCMC to explore correspondences for structure-from-motion problems, even incorporating a "jump parameter" to allow the algorithm to jump to new solutions [9]. In 2005, Barbu and Zhu proposed the SW-cut algorithm [2] which, for the first time, generalized the SW method to arbitrary probabilities models. As we will discuss in later sections, the SW-cut did not consider negative edges, high order constraints, or hierarchical graphs and is less effective in swapping between competing solutions. The C^4 algorithm in this paper is a direct generalization of the SW-cut algorithm [2].

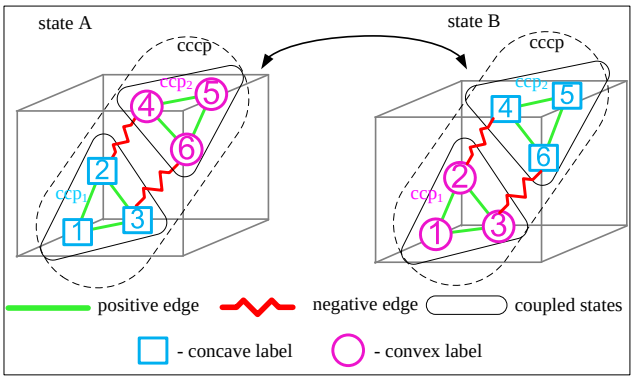


Fig. 2: Swapping between the two interpretations of the Necker Cube. Locally coupled labels are swapped with alternate labelings to enforce global consistency. See text for explanation.

1.3. Overview of the Major Concepts of C^4

In this paper we present a probabilistic clustering algorithm called *Clustering Cooperative and Competitive Constraints* (C^4) for computing multiple solutions in graphical models. We consider two types of graphs:

Adjacency graphs treat each node as an entity, such as a pixel, a superpixel, a line, or an object, which has to be labeled in K-classes (or colors). Most MRFs and CRFs used in computer vision are adjacency graphs.

Candidacy graphs treat each node as a candidate or hypothesis, such as a potential label for an entity, or a detected object instance in a window, which has to be confirmed ('on') or rejected ('off'). In other words, the graph is labeled with $K = 2$ colors.

As we will show in Section 2.1, an adjacency graph can always be converted to a bigger candidacy graph. In both cases, the tasks are posed as graph coloring problems on MRF, CRF or hierarchical graphs. There are two types of edges expressing either hard or soft constraints (or coupling) between the nodes.

Positive edges are cooperative constraints that favor the two nodes having the same label in an adjacency graph or being turned on (or off) simultaneously in a candidacy graph.

Negative edges are competitive or conflicting constraints that require the two nodes to have different labels in an adjacency graph or one node to be turned on and the other turned off in a candidacy graph.

In Fig. 2, we show that the Necker cube can be represented in an adjacency graph with each line being a node. The six internal lines are linked by 6 positive edges (in green) and two negative edges (in red and wiggly). Lines 2 and 4 have a negative edge between them as they intersect with each other, as do lines 3 and 6. We omit the labeling of the six outer lines for clarity.

In this paper, the edges play computational roles, and are used to dynamically group nodes which are strongly coupled. On each positive or negative edge, we define an *edge probability* (using bottom-up discriminative models) for the coupling strength. Then we design a *protocol* for turning these edges on and off independently according to their edge probabilities respectively for each iteration. The protocol is common for all problems while the edge probabilities are problem specific. This probabilistic procedure turns off some edges, and all the edges that remain 'on' partition the graph into some connected components (*ccp*'s).

A ccp is a set of nodes that are connected by the positive edges. For example, Fig. 2 has two *ccp*'s: ccp_1 includes nodes 1-2-3 and ccp_2 includes nodes 4-5-6. Each *ccp* is a locally coupled sub-solution.

A cccp is a composite connected component that consists of a number of *ccp*'s connected by negative edges. For example, Fig. 2 has one *cccp* containing ccp_1 and ccp_2. Each *cccp* contains some conflicting sub-solutions.

At each iteration, C^4 selects a *cccp* and updates the labels of all nodes in the *cccp* simultaneously so that (i) nodes in each *ccp* keep the same label to satisfy the positive or coupling constraints, and (ii) different *ccp*'s in the *cccp* are assigned different labels to observe the negative constraints.

Since C^4 can update a large number of nodes in a single step, it can move out of local modes and jump effectively between multiple solutions. The protocol design groups the *cccp*'s dynamically and guarantees that each step follows the MCMC requirements, such as detailed balance equations and thus it samples from the posterior probability.

We evaluate C^4 against other popular algorithms in the literature by two criteria.

1) The speed that they converge to solutions. In some studied cases, we know the global minimum solutions.
2) The number of unique solution states generated by the algorithms over time. This measures how "dynamic" an algorithm is.
3) The estimated marginal probability at each site in the graphical model after convergence.

The remainder of the paper is organized as follows: In Section 2 we describe the graph representation and an overall protocol for C^4. In Section 3 we introduce the C^4 algorithm on flat graphs and show the sampling of Potts models with positive and negative edges as a special case. In Section 4, we show experiments on generalized C^4 outperforming BP, graph cuts, SW and ICM for some segmentation, labeling, and CRF inference tasks. We extend C^4 to hierarchical graphs in Section 5 and show experiments for hierarchical C^4. Finally we conclude the paper with a discussion of our findings in Section 6.

2. Graphs, Coupling, and Clustering

2.1 Adjacency and candidacy graphs

We start with a flat graph G that we will extend to a hierarchical graph in Section 5,

$$G =< V,\ E >, \quad E = E^+ \cup E^-. \tag{1}$$

Here $V = \{v_i, i = 1, 2, ..., n\}$ is a set of vertices or nodes on which variables $X = (x_1, ..., x_n)$ are defined, and $E = \{e_{ij} = (v_i, v_j)\}$ is a set of edges which is divided into E^+ and E^- for positive (cooperative) and negative (competitive or conflicting) constraints respectively. We consider two types of graphs for G:

Adjacency graph, where each node $v_i \in V$ is an entity, such as a pixel or superpixel in image labeling, a line in a line drawing interpretation, or an object in scene understanding. Its variable $x_i \in \{1, 2, 3, ..., K_i\}$ is a label or color. MRFs and CRFs in the literature belong to this category, and the task is to color the nodes V in K colors.

Candidacy graph, where each node $v_i \in V$ is a candidate or hypothesis, such as a potential label assignment for an entity, an object instance detected by bottom-up methods, or a potential match of a point to another point in graph matching. Its variable $x_i \in \{'\text{on}', '\text{off}'\}$ is a boolean which confirms ('on') or rejects ('off') the candidate. In other words, the graph is labeled with $K = 2$ colors. In the graph matching literature [6], the candidacy graph is represented by a assignment matrix.

An adjacency graph can always be transferred to a bigger candidacy graph by converting each node v_i into K_i nodes $\{x_{ij}\}$. $x_{ij} \in \{'\text{on}', '\text{off}'\}$ represents $x_i = j$ in the adjacency graph. These nodes observe a mutual exclusion constraint to prevent fuzzy assignments to x_i.

Fig. 3 shows this conversion. The adjacency graph $G_{\text{adj}} =< V_{\text{adj}}, E_{\text{adj}} >$ has six nodes $V_{\text{adj}} = \{A, B, C, D, E, F\}$ and each has $3 \sim 5$ potential labels. The variables are $X_{\text{adj}} = (x_A, ..., x_F)$ with $x_A \in \{1, 2, 3, 4, 5\}$ and so on. We convert it to a candidacy graph $G_{\text{can}} =< V_{\text{can}}, E_{\text{can}} >$ with 24 nodes $V_{\text{can}} = \{A_1, ..., A_5, ..., F_1, ..., F_4\}$. Node A_1 represents a candidate hypothesis that assigns $x_A = 1$. The $X_{\text{can}} = (x_{A_1}, ..., x_{F_4})$ are boolean variables.

Represented by the graph G, the vision task is posed as an optimization problem that computes a most probable interpretation with a posterior probability $p(X|\ \mathbf{I})$ or an energy function $\mathcal{E}(\mathcal{X})$.

$$X^* = \arg\max p(X|\ \mathbf{I}) = \arg\min \mathcal{E}(X). \tag{2}$$

To preserve the ambiguity and uncertainty, we may compute multiple distinct solutions $\{X_i\}$ with weights $\{\omega_i\}$ to represent the posterior probability.

$$(X_i, \omega_i) \sim p(X|\ \mathbf{I}), \quad i = 1, 2, ..., K. \tag{3}$$

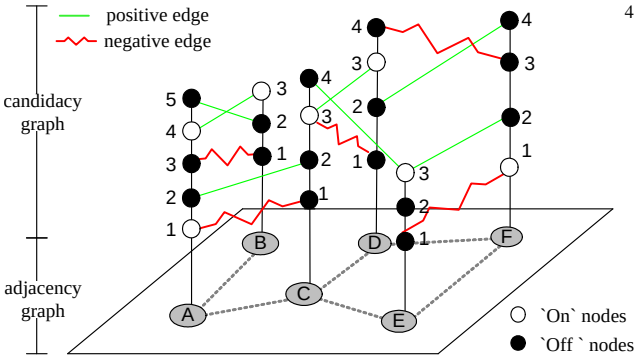


Fig. 3: Converting an adjacency graph to a candidacy graph. The candidacy graph has positive (straight green lines) and negative (wiggled red lines) edges depending on the values assigned to the nodes in the adjacency graph.

2.2 Positive and Negative Edges

In conventional vision formulation, edges in the graphs are a representational concept and the energy terms in $\mathcal{E}$ are defined on the edges to express the interactions between nodes. In contrast, Swendsen-Wang [34] and Edward-Sokal [10] added a new computational role to the edges in their cluster sampling method. The edges are turned 'on' and 'off' probabilistically to dynamically form groups (or clusters) of nodes which are strongly coupled. We will introduce the clustering procedure shortly after the example below. In this paper, we adopt this notion and the edges in graph G are characterized in three aspects:

Positive vs negative. A positive edge represents a cooperative constraint for two nodes having the same label in an adjacency graph or being turned on (or off) simultaneously in a candidacy graph. A negative edge requires the two nodes to have different labels in an adjacency graph or requires one node to be turned on and the other turned off in a candidacy graph.

Hard vs soft. Some edges represent hard constraints which must be satisfied, for example, in line drawing interpretation or scene labeling, while other edge constraints are soft and can be expressed with a probability.

Position dependent vs value dependent. Edges in adjacency graphs are generally *position dependent*. For example, in an Ising model an edge between two adjacent nodes poses a soft constraint that they should have the same label (ferromagnetism) or opposite labels (antiferromagnetism). In contrast, edges in candidacy graphs are *value dependent* and thus have more expressive power. This is common for vision tasks, such as scene labeling, line drawing interpretation, and graph matching. As Fig. 3 illustrates, the edges between nodes in the candidacy graph could be either positive or negative depending on the values assigned to nodes A, B in the adjacency graph.

As we will show in a later subsection that the positive

and negative edges are crucial for generating connected components and resolving the problem of node coupling.

2.3. The Necker Cube Example

Fig. 4 shows the construction of a candidacy graph G for interpreting the Necker cube. For clarity of discussion we assume the exterior lines are labeled and the task is to assign two labels (concave and convex) to the six inner lines such that all local and global constraints are satisfied. Therefore we have a total of 12 candidate assignments or nodes in G.

Based on the theory of line drawing interpretation [27], [33], the two 'Y'-junctions pose positive constraints so that lines 1-2-3 have the same label and lines 4-5-6 have the same label. We have 12 positive edges (green) in G to express these constraints. The intersection of lines 2 and 4 poses negative constraints that lines 2 and 4 have opposite labels which are shown in the red and wiggly edges in Fig. 4. The same is true for lines 3 and 6. The two different assignments for each line should also be linked by a negative edge. These negative edges are not shown for clarity.

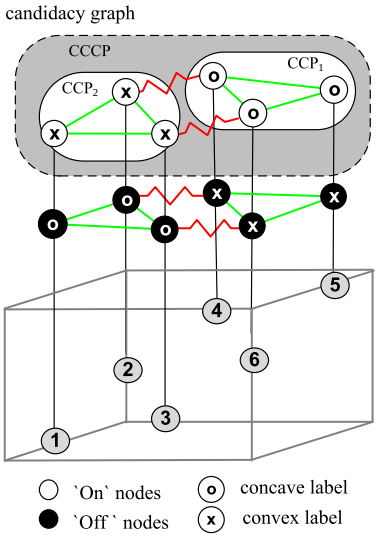


Fig. 4: The Necker cube example. The adjacency graph with 6 nodes (bottom) is converted to a candidacy graph of 12 nodes (top) for concave and convex label assignments respectively. 12 positive and 2 negative edges are placed between these candidate assignments to ensure consistency.

In this candidacy graph, the two solutions that satisfy all constraints are represented by the 2-colors in Fig. 4. The first has all nodes 1,2, and 3 labeled convex ('x') and all nodes 4,5, and 6 labeled concave ('o'). This solution is currently in the 'on' state. This would create a valid 3D interpretation where the cube is "coming out" of the page. The alternative solution has the opposite labeling, and creates a 3D interpretation of the cube "going in" to the page.

To switch from one solution to the other, we must swap the junction labels. Each set of nodes, 1-2-3 and 4-5-6, constitutes a corner of the Necker Cube and all have positive constraints between them. This indicates that we should update all of these values simultaneously. We create two *connected component* ccp_1 and ccp_2, comprised of the coupled nodes 1-2-3 and nodes 4-5-6 respectively. If we were simply to invert the labels of ccp_1 or ccp_2 alone we would create an inconsistent interpretation where all edges in the whole graph would now have the same label. What we need to do is simultaneously swap ccp_1 and ccp_2.

Notice that we have negative edges between nodes 2 and 4 and between nodes 3 and 6. Negative edges can be thought of as indicators of multiple competing solutions, as they necessarily dictate that groups on either end of the edge can either be ('on', 'off') or ('off', 'on'), creating two possible outcomes. This negative edge connects nodes in ccp_1 and ccp_2, thus indicating that those nodes in the two ccp's must have different labels. We construct a composite connected component (called $cccp$), $cccp_{12}$, encompassing nodes 1-6, we now have a full component that contains all relevant constraints. Moving from solution 1 to 2 is now as simple as flipping all the nodes simultaneously, or equivalently satisfying all of the constraints.

In the next subsection, we explain how we form the ccp's and $cccp$'s in a formal way.

2.4. Edge Probability for Clustering

On each positive or negative edge, we define an *edge probability* (using bottom-up discriminative models) for the coupling strength. That is, at each edge $e \in E$, we define an auxiliary probability $u_e \in \{0, 1\}$ or $\{'\text{on}', '\text{off}'\}$, which follows an independent probability q_e.

In Swendsen and Wang [34], the definition of q_e is decided by the energy term in the Potts model $q_e = e^{-2\beta}$ as a constant for all e. Barbu and Zhu [2], for the first time, separate q_e from the energy function and define it as a bottom-up probability: $q_e = p(l(x_i) = l(x_j)|F(x_i), F(x_j)) = p(e = on|F(x_i), F(x_j))$ with $F(x_i)$ and $F(x_j)$ being local features extracted at node x_i and x_j. This can be learned through discriminative training, for example, by logistic regression and boosting,

$$\frac{p(l(x_i) = l(x_j)|F(x_i), F(x_j))}{p(l(x_i) \neq l(x_j)|F(x_i), F(x_j))} = \sum_n \lambda_n h_n(F(x_i), F(x_j)).$$

On a positive edge $e = (i, j) \in E^+$, $u_e = 'on'$ follows a Bernoulli probability,

$$u_e \sim \text{Bern}(q_e \cdot 1(x_i = x_j)).$$

$1()$ is boolean function. It equals 1 if the condition is satisfied and 0 otherwise. Therefore, at the present state

X, if the two nodes have the same color, i.e. $x_i = x_j$, then the edge e is turned on with probability q_e. If $x_i \neq x_j$, then $u_e \sim \text{Bern}(0)$ and e is turned off with probability 1. So, if two nodes are strongly coupled, q_e should have a higher value to ensure that they have a higher probability to stay the same color.

Similarly, for negative edges $e \in E^-$, $u_e =$ 'on' also follows a Bernoulli probability,

$$u_e \sim \text{Bern}(q_e 1(x_i \neq x_j)).$$

At the present state X, if the two nodes have the same color $x_i = x_j$, then the edge e is turned off with probability 1, otherwise e is turned on with probability q_e to enforce that x_i and x_j stay in different colors.

After sampling u_e for all $e \in E$ independently, we denote the sets of positive and negative edges that remain 'on' as $E^+_{on} \subset E_+$ and $E^-_{on} \subset E^-$ respectively. Then we have a formal definitions of the *ccp* and *cccp*.

Definition 1: A *ccp* is a set of vertices $\{v_i; i = 1, 2, ..., k\}$ for which every vertex is reachable from every other vertex by the positive edges in E^+_{on}.

Definition 2: A *cccp* is a set of *ccp*'s $\{ccp_i; i = 1, 2, ..., m\}$ for which every *ccp* is reachable from every other *ccp* by the negative edges in E^-_{on}.

No two *ccp*'s are reachable by positive edges, or else they would be a single *ccp*. Thus a *cccp* is a set of isolated *ccp*'s that are connected by negative edges. An isolated *ccp* is also treated as a *cccp*.

In Section 5, we will treat the invalid cases where a *ccp* contains negative edges by converting it to a *cccp*.

To observe the detailed balance equations in MCMC design, we need to calculate the probabilities for selecting a *ccp* or *cccp* which are determined by the edge probabilities q_e. For this purpose we define their cuts. In general, a cut is the set of all edges connecting nodes between two nodes sets.

Definition 3: Under a current state X, a *cut* for a *ccp* is the set all positive edges between nodes in *ccp* and its surrounding nodes which have the same label,

$$Cut(ccp|X) = \{e : \ e \in E^+, x_i = x_j, i \in ccp, \ j \notin ccp\}.$$

These are the edges that must be turned off probabilistically (with probability $1 - q_e$) in order to form the *ccp* and the cut depends on the state X.

Definition 4: A cut for a *cccp* at a state X is the set of all negative (or positive) edges connecting the nodes in the *cccp* and its neighboring node which have different (or same) labels,

$$Cut(cccp|X) = \{e : \ e \in E^-, i \in cccp, \ j \notin cccp, x_i \neq x_j\}$$

$$\cup \{e : \ e \in E^+, i \in cccp, \ j \notin cccp, x_i = x_j\}.$$

All these edges must be turned off probabilistically with probability $1 - q_e$ in order to form the composite connected component *cccp* at state X.

As edges in E^+_{on} only connect nodes with the same label, so all nodes in a *ccp* have the same label. In contrast, all edges in E^-_{on} only connect nodes with different labels, adjacent *ccp*'s in a *cccp* must have different labels.

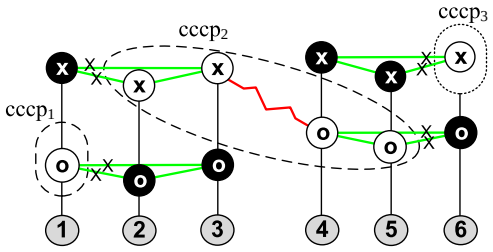


Fig. 5: A Necker cube candidacy graph not in a solution state.

To illustrate the concepts, we show a non-solution state X for the Necker cube in Figure 5. By turning off some edges (marked with the crosses), we obtain three *cccp*'s for the nodes that are currently 'on'. In this example, $q_e = 1$, as these are hard constraints that are inviolable. $cccp_1$ and $cccp_3$ have only 1 node, and $cccp_2$ has two *ccp*'s with 4 nodes. The algorithm will now arbitrarily select a *cccp* and update its values according to its constraints. If it selects either $cccp_1$ or $cccp_3$, then we are one step closer to the solution. If it selects $(cccp_2)$, then all the 4 vertex labels are swapped and we have reached a solution state and will continue to swap back and forth between the two solutions.

3 C^4 ALGORITHM ON FLAT GRAPHS

In this section, we introduce the C^4 algorithm for cluster sampling on flat graphs.

3.1 Outline of the algorithm

The C^4 algorithm works iteratively following the MCMC design. In each iteration, it generates the *cccp*'s, selects (or visits) a $cccp_o$ with a probability, and reassigns labels to its *ccp*'s such that all internal negative constraints are satisfied. As the number of *ccp*'s in $cccp_o$ grows large, the number of potential labelings will grow as well. One can remedy this situation in two ways:

1) Use a constraint-satisfaction problem (CSP)-solver to solve this smaller, easier constraint satisfaction problem within $cccp_o$.
2) Use random or heuristic sampling to find a new valid labeling.

We will use the second approach throughout this paper and the number of *ccp*'s in a $cccp_o$ is in general small, so the label assignment is not a problem. The C^4 algorithm

can be a viewed as a method that breaks a large constraint-satisfaction problem into smaller fragments in $cccp_o$ which can be satisfied locally. Then it propagates the solution through iterations.

This assignment represents a move in MCMC which is accepted by the Metropolis-Hastings step with an acceptance probability. The acceptance probability account for the probabilities for generating the $cccp$'s, selecting a $cccp_o$, assigning new labels, and the posterior probability.

In summary, we state the C^4 algorithm below.

C^4 algorithm

Input: A graph $G =< V, E >$ and posterior prob. $p(X|\ \mathbf{I})$.
Calculate the edge probability $q_e, \forall e \in E$.

//q_e is a problem specific discriminative probability.
Initialize the state $X = (x_1, x_2, ..., x_n)$.

// e.g. all nodes are turned off in a candidacy graph.
Repeat
  Denote the current X by state A.
  Step 1: Generate a $cccp_o$ at state A
    $\forall e = (i, j) \in E^+$, sample $u_e \sim \text{Bern}(q_e 1(x_i = x_j))$
    $\forall e = (i, j) \in E^-$, sample $u_e \sim \text{Bern}(q_e 1(x_i \neq x_j))$
    Generate the $\{ccp\}$ and $\{cccp\}$ based on E^+_{on} and E^-_{on}
    Select a $cccp_o$ from $\{cccp\}$ probabilistically
      // Denote the prob for selecting $cccp_o$ by $q(cccp_o|A)$.
  Step 2: Assign labels to ccp's in the $cccp$ with probability: $q(l(cccp_o = L|cccp_o, A))$.
    Denote the new X as state B.
  Step 3: Calculate the acceptance probability:
    $\alpha(A \to B) = \min(1, \frac{q(B \to A)}{q(A \to B)} \cdot \frac{p(X=B|\ \mathbf{I})}{p(X=A|\ \mathbf{I})})$.
Output: Distinct states $\{X^*\}$ with highest probabilities.

We will elaborate on the probabilities used in the algorithm in the next subsection,

3.2 Calculating the Acceptance Probability

In Markov chain design, each move between two states A and B is made reversible and observes the detailed balance equation,

$$p(X = A|\ \mathbf{I})K(A \to B) = p(X = B|\ \mathbf{I})K(B \to A). \tag{4}$$

$K(A \to B)$ is the Markov chain kernel or transition probability from A to B. In the Metropolis-Hastings design,

$$K(A \to B) = q(A \to B)\alpha(A \to B),\ \forall A \neq B. \tag{5}$$

$q(A \to B)$ is the probability for proposing state B from state A, and $\alpha(A \to B)$ is the acceptance probability,

$$\alpha(A \to B) = \min(1, \frac{q(B \to A)}{q(A \to B)} \cdot \frac{p(X = B|\ \mathbf{I})}{p(X = A|\ \mathbf{I})}). \tag{6}$$

It is easy to check that the design of proposal probability in eqn.(6) and the acceptance probability in eqn.(5) makes the kernel satisfy the detailed balance equation in (4), which in turn suffices to observe the invariance condition,

$$p(X = A|\ \mathbf{I})K(A \to B) = p(X = B|\ \mathbf{I}). \tag{7}$$

So, $p(X|\ \mathbf{I})$ is the invariant probability of the Markov chain with kernel K. Now we elaborate on the design of proposal and acceptance probabilities. The acceptance probability is determined by two ratios.

(i) The ratio $\frac{p(X=B|\ \mathbf{I})}{p(X=A|\ \mathbf{I})}$ is problem specific and is not part of our design. The posterior probability can be in general form and does not have to be modified or approximated to fit the C^4 algorithm. As states A and B only differ in their labels for nodes in $cccp_o$, it often can be computed locally if the posterior probability is a MRF or CRF.

(ii) The proposal probability ratio is completely up to our design, and it includes two parts,

$$\frac{q(B \to A)}{q(A \to B)} = \frac{q(cccp_o|B)}{q(cccp_o|A)} \cdot \frac{q(l(cccp_o) = L_A|cccp_o, B)}{q(l(cccp_o) = L_B|cccp_o, A)}.$$

$q(cccp_0|A)$ and $q(cccp_0|B)$ are the probabilities for choosing $cccp_o$ at states A and B respectively. Given the chosen composite connected component $cccp_o$, in both states A and B, the assignment of new labels is independent of the surrounding neighbors of $cccp_o$ and is often assigned by equal probability (uniform) among all valid assignments in the CSP-solver. Thus they cancel out, and we have $\frac{q(l(cccp_o)=L_A|cccp_o,B)}{q(l(cccp_o)=L_B|cccp_o,A)} = 1$.

To summarize, the key to the algorithm design is the ratio $\frac{q(cccp_o|B)}{q(cccp_o|A)}$. In single site sampling, such as Gibbs sampler, each node is a $cccp_o$ and the selection is simply a visiting scheme. In C^4, the probability for choosing $cccp_o$ at a state depends on two steps: (a) How likely it is to generate $cccp_o$ by sampling the edge probabilities q_e following the Bernoulli probability. (b) How likely it is to select $cccp_o$ from the set of formed $\{cccp\}$ in states A and B. These probabilities are hard to compute, because there are a vast amount of partitions of the graph that include a certain $cccp_o$ by turning on/off edges. A partition is a set of $cccp$'s after turning off some edges.

Interestingly, the set of all possible partitions in state A is identical to those in state B, and all these partitions must share the same cut $Cut(cccp_o)$. That is, in order for $cccp_o$ to be a composite connected component, its connections with its neighboring nodes must be turned off. Even though the probabilities are in complex form, their ratio is simple and clean due to cancellation. Furthermore, given the partition, $cccp_o$ is selected with uniform probability from all possible $cccp$'s.

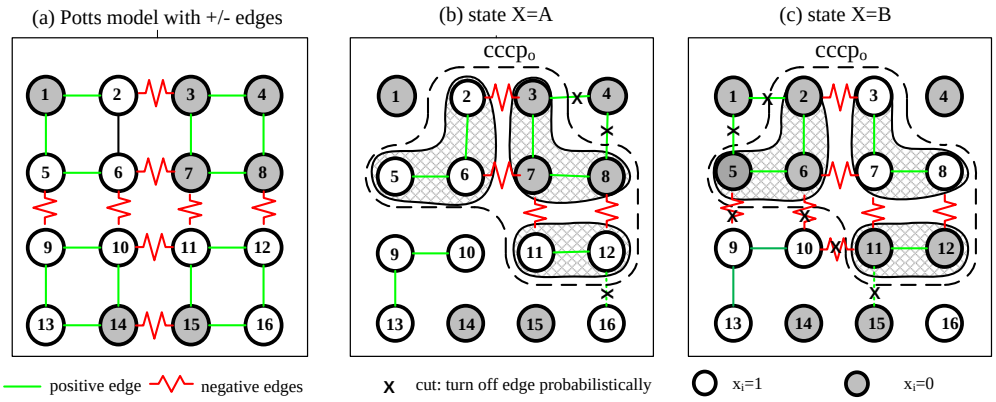


Fig. 6: The Potts model with negative edges. (a) Minimum energy is a checkerboard pattern. (b) Forming cccps. (c) $cccp_0$ consists of sub-ccps of positive edges connected by negative edges.

Proposition 1: The proposal probability ratio for selecting $cccp_o$ at states A and B is

$$\frac{q(cccp_0|B)}{q(cccp_0|A)} = \frac{\prod_{e\in Cut(cccp_o|B)}(1-q_e)}{\prod_{e\in Cut(cccp_o|A)}(1-q_e)}. \quad (8)$$

We will prove this in the appendix in a similar way to the SW-cut method [2]

3.3. Special case: Potts model with +/- edges

To illustrate C^4, we derive it in more detail for a Potts model with positive and negative edges. Let X be a random field defined on a 2D lattice with discrete states $x_i \in \{0, 1, 2, ..., L-1\}$. Its probability is specified by

$$p(X) = \frac{1}{Z}\exp\{-\mathcal{E}(X)\}; \quad (9)$$

$$\mathcal{E}(X) = \sum_{<i,j>\in E^+} \beta\delta(x_i = x_j) + \sum_{<i,j>\in E^-} \beta\delta(x_i \neq x_j),$$

where $\beta > 0$ is a constant. The edge probability will be $q_e = 1 - e^{-\beta}$ for all edges.

Fig. 6.(a) shows an example on a small lattice with $L = 2$ labels, which is an adjacency graph with position dependent edges. The states with checkerboard patterns will have highest probabilities. Fig. 6(b) and (c) show two reversible states A and B by flipping the label of a $cccp_o$ in one step. In this example, $cccp_o$ has three ccp's, $cccp_o = \{\{2, 5, 6\}; \{3, 7, 8\}; \{11, 12\}\}$. The labels of the 8 nodes are re-assigned with uniform probability, and this leads to the difference in the cuts for $cccp_o$ at the two states, $Cut(cccp_o|A) = \{(3, 4), (4, 8), (12, 16)\}$ and $Cut(cccp_o|B) = \{(1, 2), (1, 5), (5, 9), (6, 10), (10, 11), (11, 15)\}$.

Proposition 2: The acceptance probability for C^4 on the Potts model is $\alpha(A \to B) = 1$ for any two states with different labels in $cccp_o$. Therefore, the move is always accepted.

The proof follows two observations. Firstly, the energy terms inside and outside $cccp_o$ are the same for both A and B, and they differ only at the cuts of $cccp_o$. More precisely, let $c = |Cut(cccp_o|B)| - |Cut(cccp_o|A)$ be the difference of sizes in the two cuts (i.e. $c = 3$ in our example), it is not too hard to show that

$$\frac{p(X = B|\ \mathbf{I})}{p(X = A|\ \mathbf{I})} = e^{-\beta c} \quad (10)$$

Secondly, we have the proposal probability ratio, following eqn.(8),

$$\frac{q(cccp_0|B)}{q(cccp_0|A)} = \frac{(1-q_e)^{|Cut(cccp_o|B)|}}{(1-q_e)^{|Cut(cccp_o|A)|}} = e^{\beta c}. \quad (11)$$

Plugging the two ratios in eqn.6, we have $\alpha(A \to B) = 1$. In the literature of SW [10], Edwards and Sokal explain the SW on Potts model as data augmentation where the edge variables $\{u_e\}$ are treated as auxiliary variables and they sample $\{x_i\}$ and $\{u_e\}$ iteratively from a joint probability.

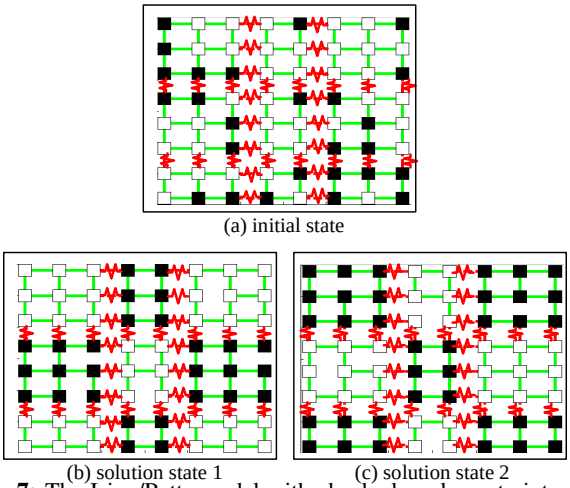


Fig. 7: The Ising/Potts model with checkerboard constraints and two minimum energy states computed by C^4.

4. Experiments on Flat Graphs

In this section we test C^4's performance on some flat graphs (MRF and CRF) in comparison with the Gibbs sampler [15], SW method [34], iterated conditional modes (ICM), graph cuts [4], and loopy belief propagation (LBP) [21]. We choose classical examples: (i) Ising/Potts model for MRF; (ii) Line drawing interpretation for constrained-satisfaction problem using candidacy graph; (iii) scene labeling using CRF; and (iv) scene interpretation of aerial images.

4.1. Checkerboard Ising Model

We first show the Ising model on a 9×9 lattice with positive and negative edges (the Ising model is a special case of the Potts model with $L = 2$ labels). We tested C^4 with two parameters settings: (i) $\beta = 1$ and thus $q_e = 0.632$; and (ii) $\beta = 5$ and thus $q_e = 0.993$. In this lattice we've created a checkerboard pattern. We've assigned negative and positive edges so that blocks of nodes want to be the same color, but these blocks want to be different colors than their neighbors.

Fig. 7 shows a typical initial state to start the algorithm, and two solutions with minimum (i.e. 0) energy. Fig. 8(a) shows a plot of energy versus time for C^4, Gibbs sampler, SW, graph cuts, and LBP. C^4 converges second fastest of all five algorithms in about 10 iterations, behind graph cuts. Belief propagation cannot converge due to the loopiness of the graph, and Gibbs sampler and the conventional Swendsen-Wang cannot quickly satisfy the constraints as they do not update enough of the space at each iteration. This shows that C^4 has a very low burn-in time.

Fig. 8(b)(c) show the state visited at each iteration. We show the states in 3 levels: the curve hits the ceiling or floor for the two minimum energy states respectively, and the middle for all other states. Here we are only comparing graph cuts, SW and C^4 as they are the only algorithms that converge to a solution in a reasonable amount of time. C^4 keeps swapping solutions while SW and graph cuts get stuck in their first solution. This is because C^4 can group along negative edges as well as positive edges to update large portions of the system at once, while Swendsen-Wang is stuck proposing low probability moves over smaller portions of the solution space.

We also compared our results for experiments where $\beta = 1$ and $\beta = 5$. Figure 8(c) shows the states visited by the sampler over time. In the $\beta = 1$ case, it takes longer for C^4 to converge, because it can't form large components with high probability. As β gets large, however, C^4 very quickly takes steps in the space towards the solution and can move rapidly between solution states. We have found that an annealing schedule where $q_e = 1 - e^{-\beta/T}$ and T is adjusted such that q_e moves from 0 to 1 over the course of the experiment works quite well too.

We finally compare the estimated marginal beliefs at each node as computed by each algorithm. LBP computes these beliefs directly, but we can estimate them for Gibbs sampling, SW, and C^4 by running each algorithm and recording the empirical mean at each iteration for each node given the previous states. Fig. 9 shows the belief for one of the Ising model sites over time for each of the 4 algorithms. LBP does not converge, so it has a noisy estimate over time and is not plotted for clarity, Gibbs and SW converge to a probability of 1, because they get stuck in a single solution state, while C^4 approaches 0.5, as it keeps flipping between the two states.

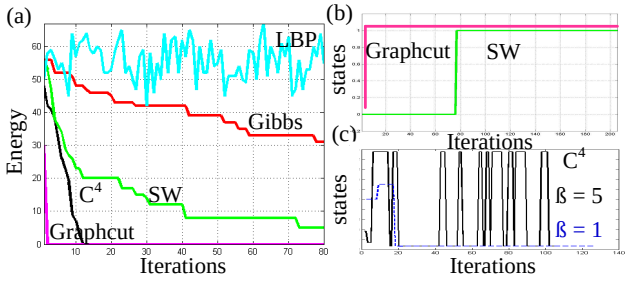


Fig. 8: (a) Energy plots of C^4, SW, Gibbs sampler, graph cuts, and LBP on the Ising model vs. time. (b) (c) The state (visited by the algorithms) in time for graph cuts, SW and C^4. Once SW and graph cuts hit the first solution, they get stuck while C^4 keeps swapping between the two minimum energy states. C^4 results shown for $\beta = 1$ and $\beta = 5$.

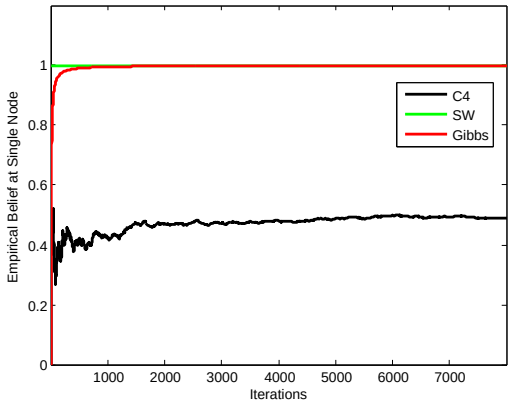


Fig. 9: Comparison of marginal beliefs at a single site of the Ising model for Gibbs, SW, and C^4. C^4 correctly converges toward 0.5, while the other algorithms only find a single solution state. LBP does not converge and thus has erratic beliefs that we do not show on this plot.

4.2. Checkerboard Potts Model with 7 Labels

We ran the same experiment as with the Ising model above but this time solved the same checkerboard pattern on a Potts model in which each site could take one of seven possible colors ($L = 7$). In this example, we have a

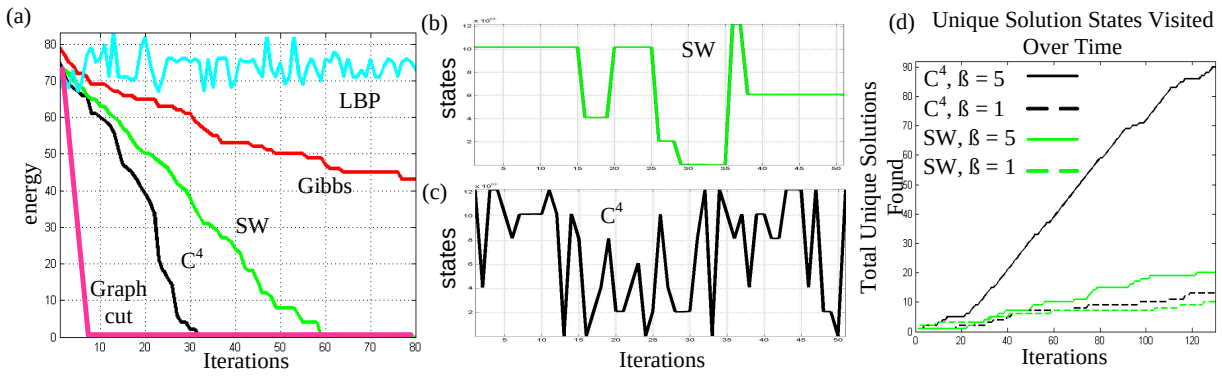


Fig. 10: (a) Energy plots of C^4, SW, Gibbs sampler, and LBP on the Potts model ($L = 7$) vs. time. (b) (c) The minimum energy states visited by SW and C^4 algorithms over time. (d) Total number of unique solutions found vs. time for SW and C^4 with $\beta = 1$ and $\beta = 5$.

large number of equal states (in checkerboard pattern) with minimum energy.

Fig. 10(a) plots the energy convergence of each algorithm over time. Graph cuts again converges to just one of the many solutions. Unlike in the case of the $L = 2$ model, SW is able to find multiple solutions this time, as seen in Fig. 10(b). Fig. 10(c) shows the number of distinct states with minimum energy that have been visited by SW and C^4 over time. We see that C^4 explores more states in a given time limit which again demonstrates that C^4 is more dynamic and thus have fast mixing time – a crucial measure for the efficiency of MCMC algorithms. We also compare the case where $\beta = 1$ vs. $\beta = 5$. Once again, we see that $\beta = 1$ doesn't create strong enough connections for C^4 to move out of local minimum, so it finds roughly as many unique solutions as Swendsen-Wang does (about 13). When β is increased to 5, however, the number skyrockets from 13 to 90. We thus see that C^4 can move around the solution space much more rapidly than other methods when β is high and can discover a huge number of unique solution states.

4.3. Line Drawing Interpretation

The previous two examples are based on MRF models whose edges are position dependent. Now we test on line drawing interpretation on candidacy graph. We use two classical examples which have multiple stable interpretations, or solutions: (i) the Necker cube in Fig. 1 that has two interpretations; and (ii) a line drawing with double cubes in Fig. 11 that has four interpretations. The swapping between these states involves the flipping of 3 or 12 lines simultaneously. Our goal is to test whether the algorithms can compute the multiple distinct solutions over time.

We adopt a Potts like model on the candidacy graph. Each line in the line drawing is a node in the Potts model, which can take one of eight line drawing labels indicating whether the edge is concave, convex, or a depth boundary. See [33] for an in-depth discussion on labels for consistent line drawings. We add an edge in our candidacy graph between any two lines that share a junction. At each junction, there are only a small set of valid labels for each line that are realizable in a 3D world. We add positive edges between pairs of line labels that are consistent with one of these junction types, and negative edges between line labels that are not. Thus, we model the pairwise compatibility of neighboring line labels given the type of junction they form.

For these experiments we set $\beta = 2$, resulting in $q_e = 0.865$. Figs 11(a) and (c) plot the state visited by the algorithms over time. Once again we see that C^4 can quickly switch between solutions where CSP solvers or other MCMC methods could get stuck.

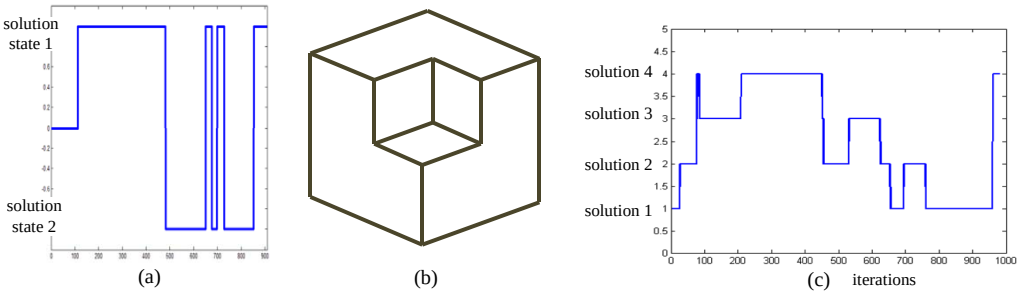
4.4. Labeling Man-made Structures on CRFs

We recreate the experiments in [22], where CRFs were learned to model man-made structures in images. Fig. 12 shows images that are broken into 16x24 grids and assigned a label $x_i = \{-1, +1\}$ if they cover man-made structure or not. The probability of labeling the sites x given data y in a Conditional random field (CRF) is

$$p(X|Y) = \frac{1}{Z} \exp \sum_i \phi(x_i, y) + \sum_i \sum_{j \in \mathcal{N}_i} \psi(x_i, x_j, y) \ . \tag{12}$$

For space we refer the reader to [22] for more details of this model. We simply choose their model so that we can compare various algorithms on the same representation. The authors use a greedy algorithm (ICM) for inference.

We learned the CRF weights via BFGS [13] using the data from [22] and compared inference results using C^4 to ICM, LBP, and SW. Edge probabilities were taken from the CRF interaction potentials. The CRF defines a potential for the case when two sites have the same label and for when they have different labels. If the ratio of these two



(a) (b) (c)

Fig. 11: Experimental results for swapping state between interpretations: (a) States visited by C^4 for the Necker Cube. (b) A line drawing with outer and inner cubes. (c) States visited by C^4 for the double cubes.

Method	FalsePositive (per image)	DetectRate (%)
SW	46.51	0.556
ICM	46.90	0.651
LBP	47.18	0.685
C^4	**47.12**	**0.696**

TABLE 1: False positives per image and detection rate using Loopy BP, SW, ICM, and C^4 for man-made structure detection.

potentials was below a threshold τ, a negative edge was used to connect the sites to enforce them to be labeled differently.

Fig. 12 shows the detection results and ground truths. LBP has very few false positives, but misses huge amounts of the detection. ICM looks graphically similar to C^4, but produces significantly more false positives. C^4 is able to swap between foreground/background fragments in large steps so it can find blocks of man-made structure more effectively.

Table 1 shows our results as in [22]. We used an implementation for that paper provided by K. Murphy[1] that achieves a higher false positive rate than Hebert's model. Nevertheless, our goal is not to outperform [22], but to show that, given the same learned model, C^4 can outperform other inference methods. Here we see that, for roughly the same false alarm rate, C^4 has a higher detection rate than other algorithms.

4.5. Parsing Aerial Images

In this experiment we use C^4 to parse aerial images. This experiment is an extension of our work from [31]. In [31], aerial images are represented as collections of groups of objects, related by statistical appearance constraints. These constraints are learned automatically in an offline phase prior to inference.

We create our candidacy graph by letting each bottom-up detected window be a vertex in the graph, connected by edges with probabilities proportional to how compatible those objects are (we refer to [31] for detailed discussion of the energy function). Each candidate can be on or off, indicating whether it is in the current explanation of the scene or not.

Each edge is assigned to be positive or negative and assigned a probability q_e of being on by examining the energy $e = \phi(x_i, x_j)$ between its two nodes. If $e > t$, the edge is labeled as a negative edge and if $e < t$ the edge is labeled as a positive edge, where t is a threshold of the user's choosing. In our experiments we let $t = 0$. In this way we create data-driven edge probabilities and determine positive and negative edge types for C^4.

In these experiments we learned a prior model for likely object configurations using labeled aerial images. Object boundaries were labeled in each image from a set of over 50 images. We tested the results on five large aerial images collected from Google Earth that were also labeled by hand, so that we could measure how much C^4 improved the final detection results. Though we only use five images, each image is larger than 1000x1000 and includes hundreds of objects, so one could also think of the evaluation as spanning 125 200x200 images.

Figure 13 shows an example of a parsed aerial scene. The bottom-up detected windows are treated as candidates and many are false positives. After using C^4 minimizing a global energy function, however, we are left with the subset that best satisfies the constraints of the system. The false positive rates are vastly diminished after C^4 rules out incompatible proposals. Figure 13(d) shows the precision-recall curve for aerial image object detection using C^4 vs. just bottom-up cues. We can see that the C^4 curve, drawn in dashed green, has a much higher precision than the bottom-up detections even as the recall increases.

We also compared the results of using C^4 over LBP, ICM, and SW for similar false alarm rates. The results are shown in Table 2

Fig. 14 shows the true power of C^4. Panel (a) shows an interpretation of a scene that C^4 initially incorrectly labeled as a line of cars in a parking lot, which it has mistaken some building vents for. Because C^4 can simultaneously swap these competing sub-solutions, however,

1. http://people.cs.ubc.ca/∼murphyk/Software/CRF/crf2D_kumarData.html

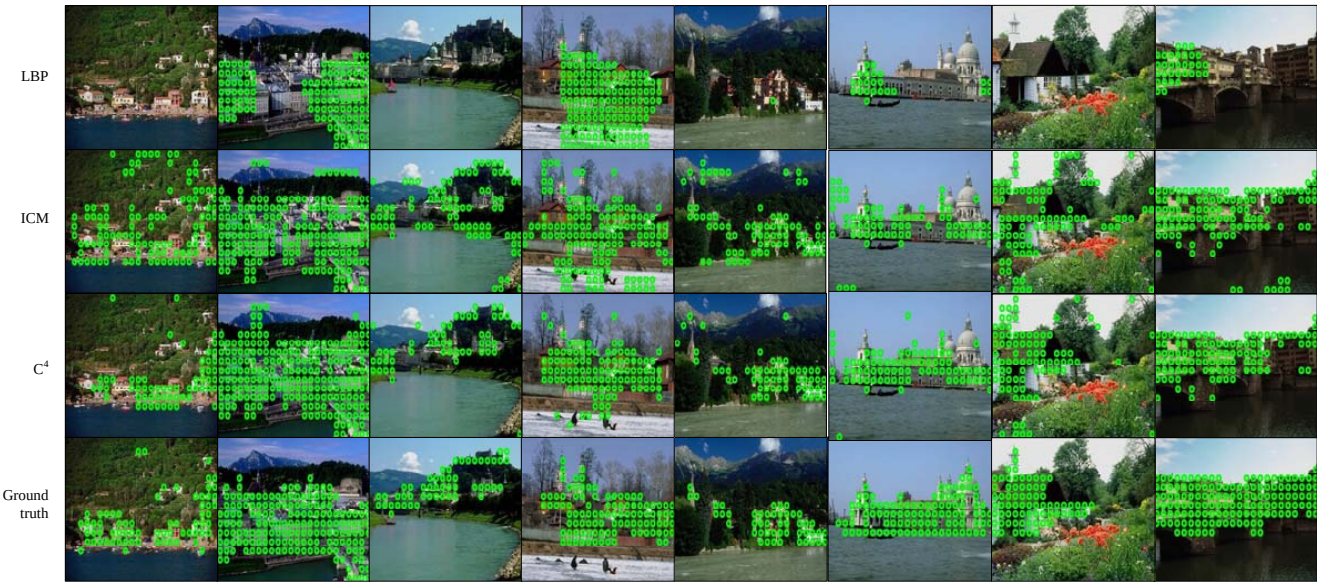


Fig. 12: Man-made structure detection results using a CRF model with LBP, ICM, and C^4.

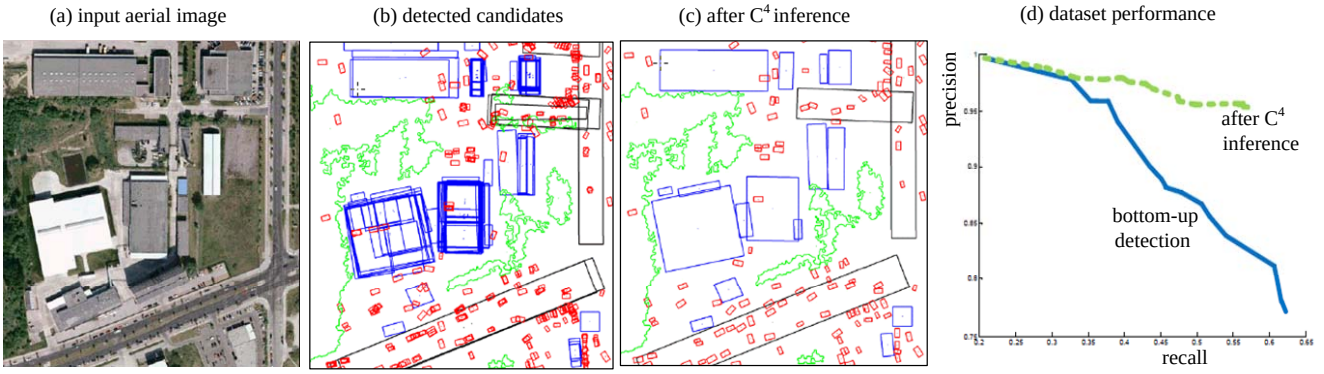
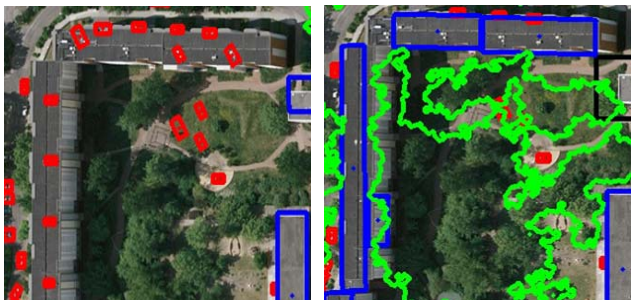


Fig. 13: An application of C^4 to aerial image parsing. (a) A portion of an aerial image from Google earth. (b) A set of bottom-up detections of objects with each being a candidate, i.e. a node in the candidacy graph. Note the large number of false positives that need to be turned off. (b) The final subset of proposals selected by C^4 to represent the scene. C^4 has removed the candidates that are inconsistent with the prior. (c) Precision recall curve for pixel level performance over a dataset of aerial images.



(a) interpretation 1: cars in parking lot (b) interpretations 2: roof with vents

Fig. 14: Alternate sub-solutions during scene parsing. C^4 swaps between these two interpretations during inference. (a) Vents on top of building roofs incorrectly detected as cars in parking lot. (b) Vents on top of buildings correctly grouped with buildings, thus removing the car detections.

Method	FalsePositive (per image)	DetectRate (%)
LBP	85.32	0.668
ICM	82.11	0.768
SW	87.91	0.813
C^4	**83.04**	**0.875**

TABLE 2: False positives per image and detection rate using Loopy BP, SW, ICM, and C^4 for aerial image parsing.

we see in Panel (b) that, later in the algorithm, C^4 has settled on the correct explanation.

5. C^4 ON HIERARCHICAL GRAPHS

In this section, we discuss the consistency of the flat graphs and extend C^4 from flat graphs to hierarchical graphs and then we address high-order constraints that involve more than two sites.

5.1 Condition for Graph Consistency

In each iteration of the C^4 algorithm, suppose we have turned on edges probabilistically and the original graph $G =< V, E >$ becomes $G_{on} =< V, E_{on} >$ with $E = E_{on} \cup E_{off}$, $E_{on} = E^+_{on} \cup E^-_{on}$, and $E_{off} = E^+_{off} \cup E^-_{off}$. As we discussed in Section 2.4 all nodes in the graph G_{on} in each *ccp* shares the same label and they are supposed to form a coupled partial solution. However, if the constraints in graph G are inconsistent, then some nodes in a *ccp* may be connected by edges in E^-_{off}. Though such negative edges are not turned on in *ccp*, they indicate that some nodes in the *ccp* may be conflicting to each other. This may not be a serious problem, for example, the negative edges may simply express soft constraints, such as overlapping windows due to occlusion, which is acceptable in the final solution.

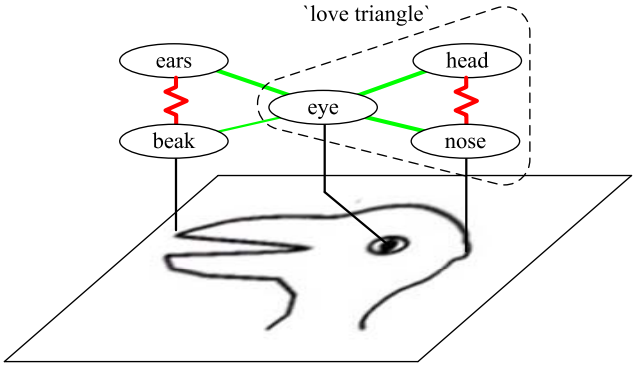


Fig. 15: An attempt to solve the duck/rabbit illusion using flat C^4. We see that we are very likely to form love triangles on the left and right of the graph, making constraint satisfaction very difficult.

Fig. 15 shows an example where the negative edge is a hard constraint. If we try to solve the duck/rabbit illusion using flat candidacy graph, a *ccp* may contain $\{'eye', 'nose', 'head'\}$ which is inconsistent. We call it a "love triangle".

Definition 5: In a graph G, two nodes i, j connected by a negative edge is said to be involved in a *love triangle* if there also exists a path between i, j that consists of all positive edges.

Definition 6: A *ccp* is said to be *consistent* in graph G if there are no negative edges in E that connect two nodes in the *ccp*, that is, $\{e : i, j \in ccp\} \cap E^- = \emptyset$. A graph G is said to be *consistent* if all its *ccp*'s are always consistent in C^4.

When a graph is consistent, then we are guaranteed to get valid solutions.

The existence of the so-called 'love triangles' are the sole reason to generate inconsistent *ccp*'s. For this we can easily prove the following proposition.

Proposition 3: In the absence of 'love triangles', the graph G will be consistent.

The essential reason for generating the 'love triangles' in a graph, mostly in candidacy graphs, is that certain nodes are over-loaded with multiple labels and thus they are coupled with conflicting nodes. For example, the node 'eye' should be either a 'rabbit eye' or a 'duck eye' and it should be split into two conflicting candidates connected by an negative edge. This way it can eliminate the "love triangle". Fig. 16 illustrates that we can remove the love triangle by splitting node 1 into nodes 1 and $1'$ and thus we will have consistent *ccp*.

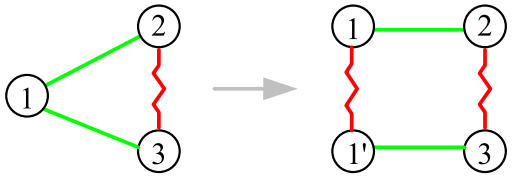


Fig. 16: Breaking the 'love triangle' in a candidacy graph.

5.2. Formulation of Hierarchical C^4

One other common issue that we need to address is higher-order constraints that involve more than 2 nodes. Fig. 17 shows a hierarchical graph representation for the duck/rabbit illusion. This is a candidacy graph with two layers. The top layer contains two hidden candidate hypotheses: 'duck' and 'rabbit'. The two nodes are decomposed into three parts in layer 1 respectively and thus impose high order constraints between them. Now the hypotheses for parts are specifically for 'duck.eye', 'rabbit.eye' etc. The negative edge connecting the two object nodes is inherited from their overlapping children.

This hierarchical candidacy graph is constructed on-the-fly with nodes being generated by multiple bottom-up detection and binding processes as well as top-down

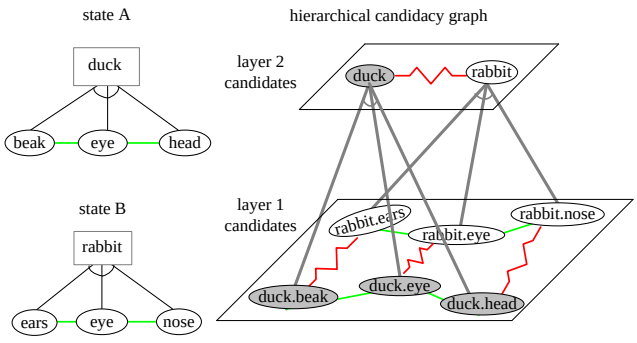


Fig. 17: An attempt to solve the duck/rabbit illusion using hierarchical C^4. The trees define which parts comprise each object. Nodes are grouped according to these trees, creating higher-level nodes. The higher-level nodes inherit the negative constraints.

prediction processes. We refer to a recent paper by Wu and Zhu [39] for the various bottom-up/top-down processes in object parsing. In this graph, positive and negative edges are added between nodes on the same layers in a way identical to the flat candidacy graph, while the vertical links between parent-child nodes are deterministic.

By turning on/off the positive and negative edges probabilistically at each layer, C^4 obtains ccp's and $cccp$'s as in the flat candidacy graphs. In this case, a ccp contains a set of nodes that are coupled in both horizontal and vertical directions and thus represents a partial parse tree. A $cccp$ contains multiple competing parse trees, which will be swapped in a single step. For example, the left panel in Fig. 17 shows two ccp's for the duck and rabbit respectively which are connected with negative edges in the candidacy graph.

This hierarchical representation can also eliminate the inconsistency caused by overloaded labels. That is, if a certain part is shared by multiple object or object instances, we need to create multiple instances as nodes in the hierarchical candidacy graph.

5.3. Experiments on Hierarchical C^4

To demonstrate the advantages of hierarchical C^4 over flat C^4, we present two experiments (i) interpreting the duck/rabbit illusion, and (ii) finding configurations of object parts amidst extremely high noise.

(i) Experiment on Hierarchical Duck/Rabbit Illusion. As referenced above, C^4 on the flat candidacy graph in Fig. 15 creates two love triangles. The top panel of Fig. 18 shows the results of flat C^4 on the duck/rabbit illusion. C^4 continuously swaps between two states, but the two states either have all nodes on or all nodes off, neither of which are valid solutions. The bottom panel of Figure 18 shows the results of applying hierarchical C^4 to the duck/rabbit illusion. We defined a tree for the duck/rabbit illusion consisting of either a duck, $\{beak, eye, duck_{head}\}$, or a rabbit $\{ears, eye, rabbit_{head}\}$. As a result, the algorithm instantly finds both solutions and then proceeds to swap between them uniformly. These results show that hierarchical C^4 can help guide the algorithm to more robust solutions and negates the effects of love triangles.

(ii) Experiments on Object Parsing. A problem that often appears in computer science is the problem of finding the optimal subset from a larger set of items that minimizes some energy function. For example, in the star model [12] , many instances of each object part may be detected in the image. However, our algorithm should find the subset (or subsets) of these detections that creates the highest probability configuration. This is a combinatorially hard problem as the number of solutions grows exponentially in the number of detections, so heuristic approaches are usually proposed to deal with this situation. One can

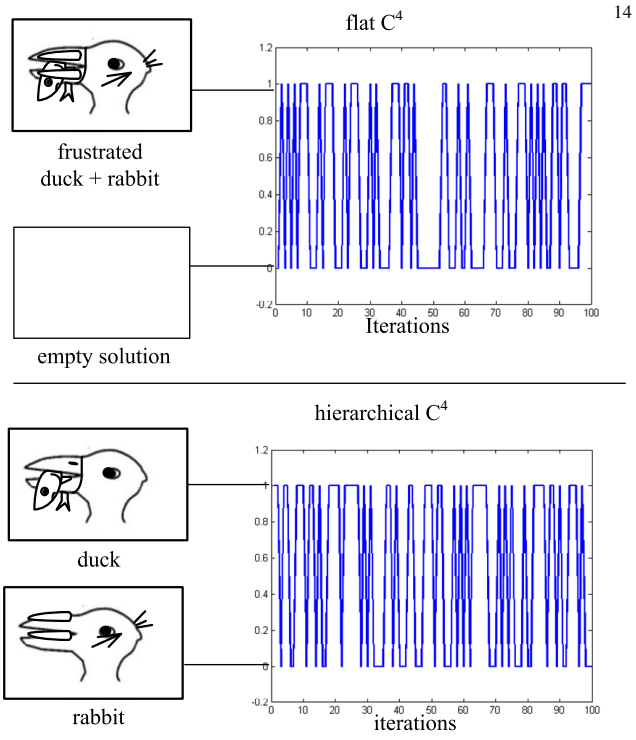


Fig. 18: (Top panel) Flat C^4 results on the duck/rabbit illusion. C^4 swaps between two impossible states due to love triangles. (Bottom panel) Hierarchical C^4 results on the duck/rabbit solution. C^4 now swaps uniformly between the two correct solutions.

use dynamic programming for inferring star models, but these require that the root part be present, which our algorithm does not. Hierarchical C^4 is ideally suited for this problem, as it can use local edge constraints and hierarchical grouping to guide its search through large sets of detections to find the most likely solutions.

In this experiment, we learned a star model for 4 categories of objects: 'side car', 'front car', 'teapot', and 'clock'. We collected 25-50 images from the Lotus Hill Dataset [38] for each of the categories, which include the true labeled parts of each object. We then added 50 false detection at random orientations, scales, and positions, for each part to serve as distractors, as shown in Figure 19. The goal is to see if the algorithms can identify the true ground truth configuration amidst a huge number of distractors. If so, then such an algorithm could thrive when we hae weak part detectors but strong geometric information. If we only considered configurations of 4 parts, finding the optimal configuration would require exhaustively searching 64,684,950 configurations, which quickly becomes intractable when considering larger configurations or more detections.

Fig. 19 shows that ICM and Swendsen-Wang find completely unreasonable solutions. Flat C^4 does not find

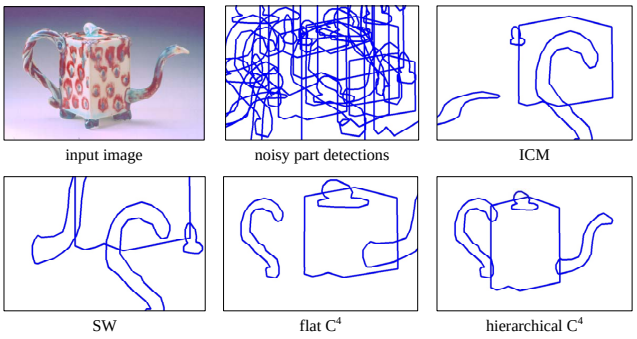


Fig. 19: Hierarchical C^4 for detecting signal from noise. A huge set of distractors are added over a true parse of an object. Using a spatial model, C^4 can find the best subset while other algorithms cannot.

the correct solution, although it does find a set of parts that look similar to a teapot. Hierarchical C^4, on the other hand, quickly converges to the true solution amidst the myriad other possible part combinations available.

Figure 21 shows the results of other signal-from-noise images that were generated as above. We show the results divided roughly into three categories: good, medium, and bad results. We see that hierarchical C^4 gets mostly good results, while ICM gets entirely bad results.

Figure 22 shows the energy of the system over time for the four algorithms we tested. Not only does hierarchical C^4 achieve a minimum energy almost instantaneously, but that both hierarchical C^4 and flat C^4 are able to achieve lower energy minimums than the other methods. This improvement applies to graph cuts as well, which, as mentioned, are not shown here because no implementation we found was able to converge in the presence of love triangles. This result shows Hierarchical C^4's ability to quickly find deeper energy minima than competing approaches.

We also tested our negative edge selection criteria. We use a threshold the pairwise probabilities computed by the star model to create negative and positive edges. We had empirically arrived at a threshold between positive and negative edges of 1e-7. Figure 20 shows the true positive, false positive, true negative, and false negative rates for different thresholds (on a log scale). We can see that 1e-7 is a good cutoff probability as it produces a clear peak in the plots (note that all individual edge probabilities are quite low in our star models). We propose to look at general heuristics for negative edge creation in future work.

These results show the power of Hierarchical C^4 for quickly finding minimal energy subsets and swapping between equally or nearly-equally likely solutions once found, where as similar methods (Swendsen-Wang, ICM, Graph Cuts) fail to even find a viable solution.

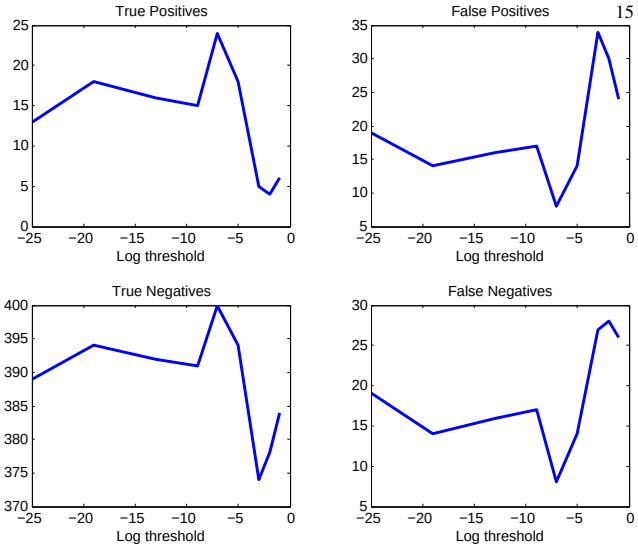


Fig. 20: True positive, false positive, true negative, and false negative rates for object part detection on the teapot category when using different negative edge thresholds. The plots all have a minimum/maximum near 1e-7.

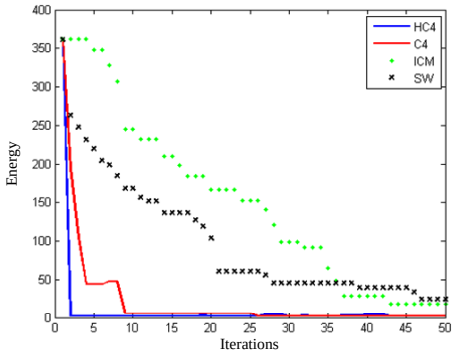


Fig. 22: Plots of energy over time for Hierarchical C^4, Flat C^4, Swendsen-Wang cuts, and iterative conditional modes (ICM). Not only does Hierarchical C^4 converge fastest of all the algorithms, but it achieves a lower energy than the other methods.

6. Discussion

In this paper we presented C^4, an algorithm that can handle complex energy minimization tasks with soft and hard constraints. By breaking a large CSP into smaller sub-CSPs probabilistically, C^4 can quickly find multiple solutions and switch between them effectively. This combination of cluster sampling and constraint-satisfaction techniques allows C^4 to achieve a fast mixing time, out-performing single-site samplers and techniques like belief propagation on existing problems. This novel algorithm can sample from arbitrary posteriors, and is thus applicable to general graphical models, including MRFs and CRFs. In addition, we were able to use a hierarchical prior to guide our search to avoid frustrations in the graph and thus achieve richer

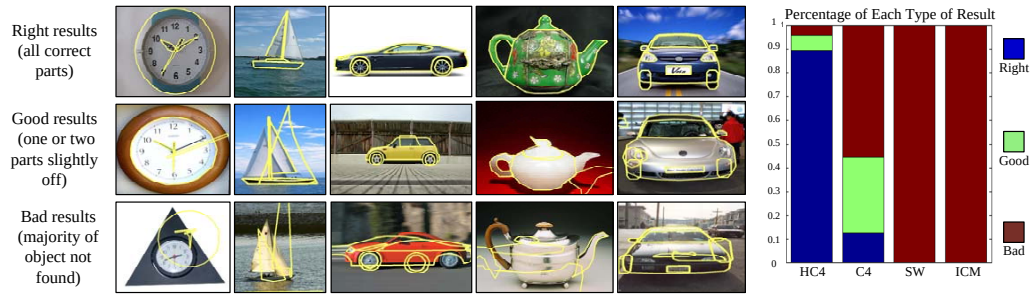


Fig. 21: Examples of good/medium/bad results for Hierarchical C^4, Flat C^4, Swendsen-Wang cuts, and iterative conditional modes (ICM). The graphs to the right show the proportion of the testing images that belonged to each ranking according to algorithm.

and more accurate results than just by using Flat C^4 alone.

In this paper, we applied C^4 to a number of simple application for illustration purpose. In two related papers by the author's group, the C^4 algorithm was applied to layered graph matching [24] and aerial image parsing [31] with state-of-the-art results. In ongoing work, we are extending C^4 for scene labeling, integrating object parsing and scene segmentation.

ACKNOWLEDGMENTS

This work was partially supported by NSF IIS grant 1018751 and ONR MURI grant N000141010933. We would also like to acknowledge the support of the LHI dataset [38]. Work done at LHI was supported by 863 grant 2009AA01Z331 and NSFC 90920009.

REFERENCES

[1] K.R. Apt, "The Essence of Constraint Propagation", *Theoretical Computer Science*, 1999.

[2] A. Barbu and S.C. Zhu, "Generalizing Swendsen Wang to Sampling Arbitrary Posterior Probabilities", *IEEE Trans. on PAMI*, vol.27, no.8, 1239-1253, 2005.

[3] J. Besag, "On the Statistical Analysis of Dirty Pictures", *J. of the Royal Stat. Soc.*, B-48:259, 302, 1986.

[4] Y. Boykov, O. Veksler and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts", *IEEE trans. on PAMI*, vol. 23, no. 11, pp. 1222-39, 2001.

[5] A. Braunstein, M. Mzard, and R. Zecchina, "Survey propagation: An algorithm for satisfiability", Random Structures Algorithms, 27 , pp. 201226, 2005.

[6] H. Chui and A. Rangarajan, "A New Point Matching Algorithm for Non-rigid Registration", *Computer Vision and Image Understanding*, 89(2): 114-141, 2003.

[7] C. Cooper and A. Frieze, Mixing Properties of the Swendsen- Wang Process in Classes of Graphs, *Random Structures and Algorithms*, vol. 15, no. 3-4, pp. 242-261, 1999.

[8] T. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, 2001.

[9] F. Dellaert, S. Seitz, C. Thorpe, S. Thrun, *Feature Correspondence: A Markov Chain Monte Carlo Approach*, Advances in Neural Information Processing Systems, vol. 13, pp. 852-858, 2001.

[10] R. Edwards, A. Sokal, "Generalization of the Fortuin-Kasteleyn-Swendsen-Wang Representation and Monte Carlo Algorithm", *Physical Rev. Letters*, vol. 38, pp. 2009-2012, 1988.

[11] P. F. Felzenszwalb and J. D. Schwartz, Hierarchical Matching of Deformable Shapes, *Proc. of IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[12] R. Fergus, P. Perona, and A. Zisserman. "A Sparse Object Category Model for Efficient Learning and Exhaustive Recognition", *Proc. of IEEE Conf. on CVPR*, 2005.

[13] R. Fletcher, "A New Approach to Variable Metric Algorithms", *Computer Journal*, 1970.

[14] A. Gelman, J. B. Carlin, H.S. Stern, D.B. Rubin, *Bayesian Data Analysis*, Second Edition, Chapman & Hall/CRC, Chapter 5, 2004.

[15] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images", *IEEE Trans. on PAMI*, vol. 6, pp. 721-741, 1984.

[16] V. K. Gore, M. R. Jerrum, "The Swendsen-Wang process does not always mix rapidly", *Proceedings of the Twenty-Ninth Annual ACM Symposium on theory of Computing* , STOC '97, ACM, 674-681, 1997.

[17] P. Green, " Reversible Jump Markov chain Monte Carlo Computation and Bayesian Model Determination", *Biometrika*, vol. 82, pp. 711–732, 1995.

[18] U. Grenander and M.I. Miller, "Representationsof Knowledge in Complex Systems", *J. Royal Statistical Soc. Series B*, vol. 56, no. 4, 1994.

[19] D. A. Huffman, "Impossible Objects as Nonsense Sentences", *Machine Intelligence*, 8:475-492, 1971.

[20] V. Kolmogorov and C. Rother, "Minimizing Nonsubmodular Functions with Graph Cuts-A Review", *IEEE Trans. on PAMI*, vol. 29, no. 7, pp. 1274-79, 2007.

[21] M. Kumar and P. Torr, "Fast Memory-Efficient Generalized Belief Propagation", *Lecture Notes in Computer Science*, 2006.

[22] S. Kumar and M. Hebert, "Man-Made Structure Detection in Natural Images using a Causal Multiscale Random Field", *Proc. of IEEE conf. on CVPR*, 2003.

[23] J. Lafferty and F. Pereira, "Conditional Random fields: Probabilistic models for segmenting and labeling sequence data", *Proc. of Int'l Conf. on Machine Learning*, 2001.

[24] L. Lin, K. Zeng, X. B. Liu and S.C. Zhu, "Layered Graph Matching by Composite Clustering with Collaborative and Competitive Interactions", *Proc. of IEEE Conf. on CVPR*, 2009.

[25] J. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer, 2001.

[26] J. Liu, W.H. Wong and A. Kong, "Correlation structure and convergence rate of the Gibbs sampler with various scans", *J. of the Royal Stat. Soc.*, vol. B 57, pp. 157-169, 1995.
[27] A.K. Mackworth, "Interpreting pictures of polyhedral scenes", *Artificial Intelligence*, 4(2):121-137, 1973.
[28] A.K. Mackworth, "Consistency in networks of relations", *Artificial Intelligence*, 8:99-118, 1977.
[29] S. Oh, J. Rehg, T. Balch, and F. Dellaert, "Learning and Inference in Parametric Switching Linear Dynamical Systems", *Proc. Int'l Conf. on Computer Vision*, v. 2, pp. 1161-1168, 2005.
[30] J. Pearl, *Heuristics:Intelligent Search Strategies for Computer Problem Solving*, 1984.
[31] J. Porway, K. Wang, and S.C. Zhu, "A Hierarchical and Contextual Model for Aerial Image Understanding", Int'l Journal of Computer Vision, vol.88, no.2, pp 254-283, 2010.
[32] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene Labeling by Relaxation Operations", *IEEE Trans. on SMC* No. 6, pp. 420-433 June 1976.
[33] K. Sugihara, "Machine Interpretation of Line Drawings", *MIT Press*, 1986.
[34] R.H. Swendsen and J.S Wang, "Nonuniversal Critical Dynamics in Monte Carlo Simulations", *Physical Review Letters*, vol. 58, no. 2, pp. 86-88, 1987.
[35] Z. Tu and S.C. Zhu, "Image Segmentation by Data-Driven Markov Chain Monte Carlo", *IEEE Trans. on PAMI*, vol. 24, pp 657–673, 2002.
[36] A. Torralba, K. Murphy, W. Freeman, "Object Detection", *Proc. of IEEE Conf. on CVPR*, 2004.
[37] Y. Weiss, "Correctness of Local Probability Propagation in Graphical Models with Loops". *Neural Computation*, 12 (1): 141, 2000.
[38] B. Yao, M. Yang, and S.C. Zhu, "Introduction to a Large-Scale General Purpose Ground Truth Database: Methodology, Annotation Tools and Benchmarks", *Proc. of Int'l Conf on EMMCVPR*, 2007.
[39] T.F. Wu and S.C. Zhu, "A Numeric Study of the Bottom-up and Top-down Inference Processes in And-Or Graphs", *Int'l J. of Computer Vision*, (accepted) 2010. (A short version appeared in ICCV09).
[40] S.C. Zhu and D. Mumford, "A Stochastic Grammar of Images", *Foundations and Trends in Computer Graphics and Vision*, Vol.2, No.4, pp 259-362, 2006.

Jake Porway received a BS degree in Computer Science from Columbia University in 2000 with a focus on intelligent systems, then completed his MS and PhD degrees in Statistics from UCLA in 2005 and 2010. During his PhD career, he worked in the CIVS lab at UCLA, where his research focused on probabilistic grammar models for object recognition in images and video, Bayesian methods for inference in graphical models, and unsupervised and semi-supervised methods for automated learning. At the time of this publication he is working as the data scientist for the New York Times R&D division and is examining the role of "big data" in modern machine learning applications.

Song Chun Zhu received the BS degree from the University of Science and Technology of China in 1991 and the MS and PhD degrees from Harvard University in 1994 and 1996, respectively. He is currently a professor with the Department of Statistics and the Department of Computer Science at the University of California, Los Angeles (UCLA). Before joining UCLA, he was a postdoctoral researcher in the Division of Applied Math at Brown University from 1996 to 1997, a lecturer in the Department of Computer Science at Stanford University from 1997 to 1998, and an assistant professor of computer science at Ohio State University from 1998 to 2002. His research interests include computer vision and learning, statistical modeling, and stochastic computing. He has published more than 100 papers in computer vision. He has received a number of honors, including the David Marr Prize in 2003 with Z. Tu et al, the J.K. Aggarwal prize from the Int'l Association of Pattern Recognition in 2008, the Marr Prize honorary nominations in 1999 and 2007 with Y.N. Wu et al, a Sloan Fellowship in Computer Science in 2001, a US National Science Foundation Early Career Development Award in 2001, and an US Office of Naval Research Young Investigator Award in 2001. In 2005, he founded, with friends, the Lotus Hill Institute for Computer Vision and Information Science in China as a nonprofit research organization (www.lotushill.org). He is a Fellow of IEEE.