Finding modal states in a discrete, sparsely connected, loopy graph by Iterated Cutset Conditioning

Gabi Teodoru, Maneesh Sahani
Gatsby Computational Neuroscience Unit

Abstract:

To find all of the global maxima of a discrete loopy graph, we iteratively eliminate non-maximal states until only the maximal ones remain; each elimination is feasibly performed by running MAP on a subset of variables for each possible configuration of its Markov boundary.

Problem:

- MAP inference in graphical models with discrete random variables:
  - Find all $\mathbf{x}^* \in \arg\max_{\mathbf{x}} P(\mathbf{x})$, $P(\mathbf{X}) \propto \prod_l f_l(\mathbf{X}_{C_l})$, $x_i \in \mathcal{X}_i$,

    where $\mathcal{X}_i$ is the domain of the ith variable and C_l represents the cliques
- NP-hard

Characteristics of our proposed solution:

- PRO:
  - Finds ALL global maxima
  - Exact solution in a loopy graph
  - Empirical runtime similar to that of other state-of-the-art algorithms
  - Empirical running time scales polynomially with the size of the graph
- CON:
  - Empirical running time scales exponentially with the number of distinct values of a clique; this implies a need for low variable cardinalities and sparsely connected graphs
  - Requires a schedule to be provided (different for each type of graph); choice of schedule greatly influences performance; we present an efficient schedule only for grids
  - Unclear why the algorithm is feasible in practice

Main ideas of Iterated Cutset Conditioning:

1. Binary mask functions
   - Let $g_k(\mathbf{x}) \in \{0,1\}$, $g_k(\mathbf{x}^*) = 1$
   - Construct probability distributions: $P_k(\mathbf{x}) \propto P_{k-1}(\mathbf{x}) g_k(\mathbf{x})$, where $P_0(\mathbf{x}) = P(\mathbf{x})$
   - Algorithm terminates when $P_k(\mathbf{x})$ assigns the same mass to all surviving states; these surviving states are the global maxima

2. Cutset conditioning (for constructing $g_k(\mathbf{x})$)
   - Let $\mathbf{X}_S$ be some set of variables, $\mathbf{X}_{B(S)}$ be its Markov boundary, and $\breve{S} = S \cup B(S)$
   - Let $M_k^*(\breve{S}_k)$ be a set of configurations $\{\mathbf{x}_S^*, \mathbf{x}_{B(S)}\}$, where $\mathbf{x}_S^* \in \arg\max_{\mathbf{x}_S} P_{k-1}(\mathbf{x}_S \mid \mathbf{x}_{B(S)})$
   - $g_k(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x}_{\breve{S}} \in M_k^*(\breve{S}_k) \\ 0, & \text{otherwise} \end{cases}$ Properties $g_k(\mathbf{x}) \in \{0,1\}$, $g_k(\mathbf{x}^*) = 1$ hold.

   (Computational considerations):
   - Only the values of $\mathbf{x}$ where $g_k(\mathbf{x}) = 1$ are stored, and not $g_k(\mathbf{x})$ itself
   - If $\breve{S}_j \subset \breve{S}_k$, $M_k^*(\breve{S}_k)$ can be computed from $M_j^*(\breve{S}_j)$; implicit dynamic programming
   - $\mathbf{x}_S^* = \arg\max_{\mathbf{x}_S} P_{k-1}(\mathbf{x}_S \mid \mathbf{x}_{B(S)})$ computed for configurations of $\mathbf{x}_{B(S)}$ not yet eliminated

3. Constraint propagation:
   - $P_k(\mathbf{x}) \propto P(\mathbf{x}) \prod_{j \le k} g_j(\mathbf{x})$
   - If $g_i(\mathbf{x}) = 1$ and $g_j(\mathbf{x}) = 0$ for some $\mathbf{x}$, then setting $g_i(\mathbf{x}) = 0$ will leave $P_k(\mathbf{x})$ unchanged
   - Applying the above idea to the sets $M_k^*(\breve{S}_k)$ yields:

$$M_j^*(\breve{S}_j) \leftarrow \left\{ \mathbf{x}_{\breve{S}_j} \mid \mathbf{x}_{\breve{S}_j} \in M_j^*(\breve{S}_j), \mathbf{x}_{\breve{S}_j \cap \breve{S}_k} \in M_j^*(\breve{S}_j \cap \breve{S}_k) \cap M_k^*(\breve{S}_j \cap \breve{S}_k) \right\}$$

   - It is possible to link the various sets $M_k^*(\breve{S}_k)$ so that removing even one configuration from one such set can trigger a chain reaction that will remove a large number of configurations from some or all of the subsets.
   - With intelligent scheduling, the links between subsets will create a connected graph, and constraints will automatically propagate on this graph between vertices until no more configurations can be removed.

Putting it all together:

1. One iteration of the algorithm:

function ProcessFactor$(\breve{S}_{prev}, \breve{S}_k, neighboring_sets)$

$M_k(\breve{S}_k) \leftarrow M_{prev}^*(\breve{S}_{prev}) \times \mathcal{X}_{\breve{S}_k \setminus \breve{S}_{prev}}$

$M_k^*(\breve{S}_k) \leftarrow \left\{ \left(\mathbf{x}_{S_k}^*, \mathbf{x}_{B(S_k)}\right) \in M_k(\breve{S}_k) \mid \mathbf{x}_{S_k}^* \in \arg\max_{\mathbf{x}_{S_k}} P_{k-1}(\mathbf{x}_{S_k} \mid \mathbf{x}_{B(S_k)}) \right\}$

for all $\breve{S}_{neighbor} \in neighboring\_sets$ **do**

link $M_k^*(\breve{S}_k)$ and $M_{neighbor}^*(\breve{S}_{neighbor})$ and **run constraint propagation**

2. The entire algorithm:

do ProcessFactor (…) with parameters supplied by the schedule **until** $\breve{S}_k$ is the set of all variables

Schedule to run it on an MRF:

- Start with subsets of 2x2 squares, and iterate through them going left to right, top to bottom
- Once all such squares have been iterated, go on to 3x3 squares, 4x4 squares, and so on
- Heuristic: skip from 4x4 to 6x6 or 7x7, etc. directly if many configurations have been eliminated
- Link each square to their immediately neighboring squares only; can show that nothing is gained by connecting to more squares.
- Algorithm stops after a square the size of the entire grid is reached
- In practice, when increasing the square size, one variable is added at one time, as shown on the right-hand side of Figure 1.

Note that better schedules may exist for this model.

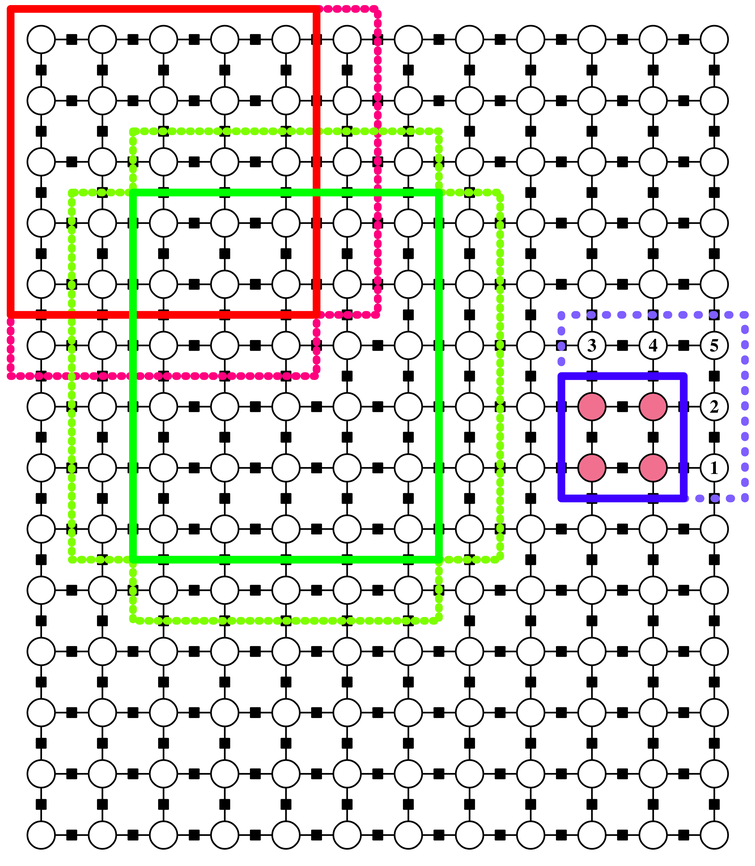

Figure 1: Top-left corner: two sets of variables (red and green) and their Markov boundaries (fuchsia and lime); Right side: The order variables are added to the set

Experiments:

We ran ICC on square-lattice MRFs of various grid sizes, with variable cardinalities of 2 and 3 and weights drawn from the following two regimes:

Regime 1: $f(x_i, x_j) \sim \text{Uniform}(-1, 1)\ \forall i \neq j, x_i, x_j$

Regime 2: $f(x_i, x_j) = \xi_{ij}(2\delta_{x_i = x_j} - 1)\ \forall i \neq j, \xi_{ij} \sim \text{Uniform}(-1,1)\ \forall i \neq j, x_i, x_j$

The number of possible configurations the algorithm keeps track of does not grow exponentially as might be expected (Fig. 2). Regime 2 (Fig. 2b) allows multiple solutions, indicated by values higher than 1 at the last iteration; in one case, there were as many as 2304 global optima.

Figure 2: Mean number of configurations at each iteration

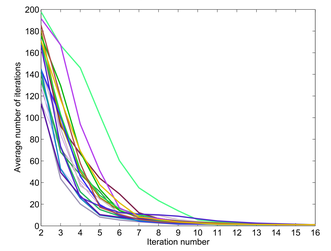

Figure 2a: Regime 1, cardinality of 3, 16x16 grid

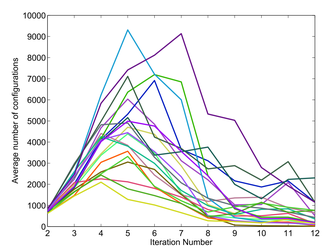

Figure 2b: Regime 2, cardinality of 3, 12x12 grid

We compared ICC with two state-of-the-art algorithms, MPLP(Sontag et al., 2008) and RPM(Schraudolph et al., 2010) on graphs using both regimes (Fig.3). Note some additional important distinctions between the algorithms:

- Variable Cardinality: RPM(2), MPLP(any), RPM(any, but runtime increases exponentially)
- Additional parameters: MPLP(none), RPM(planar embeddings of graph), ICC(schedule)
- Graph size: The particular implementation of MPLP we used was not able to handle grid sizes larger than about 30 or distributions from Regime 2

Figure 3: Comparison of algorithms' runtime on log-log graph: Regime 1: MPLP (blue), RPM(green), ICC (red); Regime 2: ICC (magenta). Solid line represents the mean and the dotted lines represent the 5% and 95% quantiles over 20 trials.

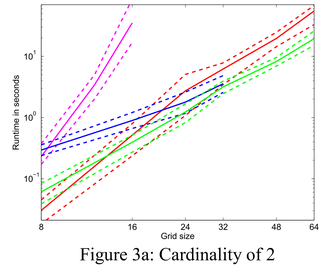

Figure 3a: Cardinality of 2

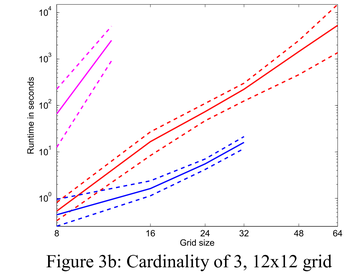

Figure 3b: Cardinality of 3, 12x12 grid

Conclusion and further work:

- Algorithm finds all global maxima.
- Main ideas fundamentally different from those of state-of-the-art exact MAP algorithms; future work will look at combining ideas from all of these algorithms

References:

- Pearl, J. (1985/1988) A Constraint-Propagation Approach to Probabilistic Reasoning. In Kanal, L.N. and Lemmer, J.F. (eds.), *UAI '85: Proceedings of the First Annual Conference on Uncertainty in Artificial Intelligence*, pp. 357-370. Rome, New York, USA: Elsevier.
- Sontag, D., Meltzer, T., Globerson, A.,Weiss, Y. & Jaakkola, T. (2008) Tightening LP Relaxations for MAP using Message Passing. *Uncertainty in Artificial Intelligence (UAI)*:503-510.
- Schraudolph, N. N. (2010) Polynomial-Time Exact Inference in NP-Hard Binary MRFs via Reweighted Perfect Matching. *Intl. Conf. Artificial Intelligence and Statistics (AISTATS)*:717-724.