Truncating the loop series expansion for BP

Vicenç Gómez[1,*]   Joris M.Mooij [2]   Hilbert J.Kappen [2]

[1]Departament Tecnologies de la Informació i de les Comunicacions
Universitat Pompeu Fabra, Barcelona, Spain
* *Visiting Radboud University*

[2]Department of Biophysics
Radboud University, Nijmegen, The Netherlands

Outline

Outline

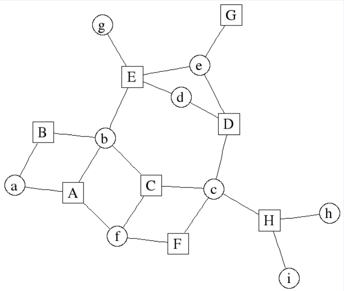
1. Introduction
   - Statistical Inference
   - Loop Calculus
2. Truncating Loop Expansion for BP
   - Loop Characterization
   - The TLSBP algorithm
   - Experimental Results
     - Ising Model
     - Medical Diagnosis
3. Conclusions

Consider a probability model on a set of *binary* variables $x_i = \pm 1, i = 1, \ldots, n$, which factorizes into m factors:

$$P(x) = \frac{1}{Z} \prod_{\alpha=1}^{m} f_\alpha(x_\alpha), \qquad Z = \sum_x \prod_{\alpha=1}^{m} f_\alpha(x_\alpha)$$

Statistical inference

- Find **marginal probabilities**, given evidence about other variables
- NP-Hard
- Message passing algorithms
- **Factor graph** unifying representation [Kschischang et al. '01]



$$P(x) = \frac{1}{Z} f_A(x_a, x_b, x_f) f_B(x_a, x_b) f_C(x_b, x_c, x_f) f_D(x_c, x_d, x_e) f_E(x_b, x_d, x_e, x_g) f_G(x_e) f_H(x_c, x_h, x_i)$$

- LDPC codes [Gallager '63], Bayes Nets. $\pi\lambda$ algorithm [Pearl '88]
- On a poly-tree (no loops), message-passing from leaves to root:

$$\begin{aligned} \text{from variable } i \text{ to factor } \alpha: \quad & n_{i\alpha}(x_i) = \prod_{\beta \ni i \setminus \{\alpha\}} m_{\beta i}(x_i), \\ \text{from factor } \alpha \text{ to variable } i: \quad & m_{\alpha i}(x_i) = \sum_{x_{\alpha \setminus \{i\}}} f_\alpha(x_\alpha) \prod_{j \in \alpha \setminus \{i\}} n_{j\alpha}(x_j), \end{aligned}$$

- After one message update, marginals can be obtained:

$$b_i^{BP}(x_i) \propto \prod_{\alpha \ni i} m_{\alpha i}(x_i)$$
$$b_\alpha^{BP}(x_\alpha) \propto f_\alpha(x_\alpha) \prod_{i \in \alpha} n_{i\alpha}(x_i)$$

- If graph has no cycles $\rightarrow$ **exact** inference ($b_i^{BP}(x_i) = P_i(x_i)$)
- If loops are present:
  - Iterative-BP (IBP) or Loopy-BP (LBP) gives an **approximation**
  - But convergence is not guaranteed

Introduction

Loop corrections

- Region-based message passing algorithms
- GBP [Yedidia et al.'01] generalizes CVM and other methods
- Cost exponential in the region size
- Choosing good regions is hard [Welling et al.'05]

Can we look for loop corrections to the BP solution?

Outline

Loop Calculus

Loop Series expansion, [Chertkov M. & Chernyak V. '06a]

Loop Series for the Factor Graph model and binary variables

$$Z = Z_{\mathrm{BP}} \left(1 + \sum_{C \in \mathcal{C}} r(C) \right) \qquad r(C) = \prod_{i,\alpha \in C} \mu_i(C) \mu_\alpha(C)$$

where:

$\log Z_{BP}$ is the Bethe-Peierls free energy (BP solution),

C are **generalized loops** (paths that have no loose ends),

$$\mu_i(C) = \frac{(1-m_i)^{q_i(C)-1} + (-1)^{q_i(C)} (1+m_i)^{q_i(C)-1}}{2(1-m_i^2)^{q_i(C)-1}}, \quad \mu_\alpha(C) = \sum_{x_\alpha} b_\alpha^{BP}(x_\alpha) \prod_{i \in C, i \in \alpha} (x_i - m_i),$$

$q_i(C)$ number of neighbors of i within the loop C, $\qquad m_i = \sum_{x_i} b_i^{BP}(x_i) x_i$

The number of loops is enourmous!

Can we truncate the loop series in some *smart* way?

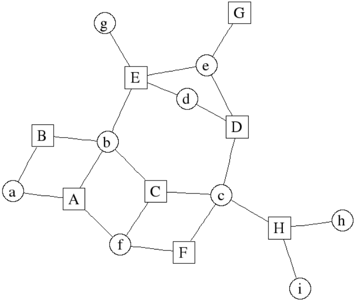
Outline

Loop Characterization

Three categories of loops: simple, branching, and complex

Generalized loop: Any subgraph C of $\mathcal{G}$ such that each node in C has degree two or larger

simple-loop (or cicuit) A Generalized Loop where all nodes have exactly degree two

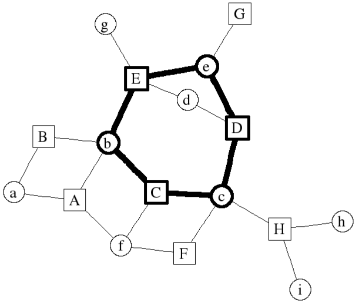


Loop Characterization

Three categories of loops: simple, branching, and complex

Generalized loop: Any subgraph C of $\mathcal{G}$ such that each node in C has degree two or larger

simple-loop (or cicuit) A Generalized Loop where all nodes have exactly degree two



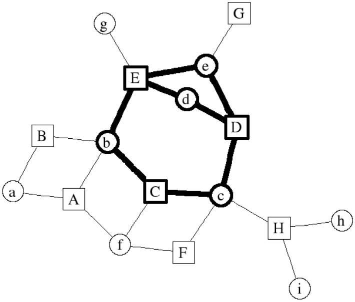
Example of simple-loop

Loop Characterization

Three categories of loops: simple, branching, and complex

Generalized loop: Any subgraph C of $\mathcal{G}$ such that each node in C has degree two or larger

simple-loop (or cicuit) A Generalized Loop where all nodes have exactly degree two



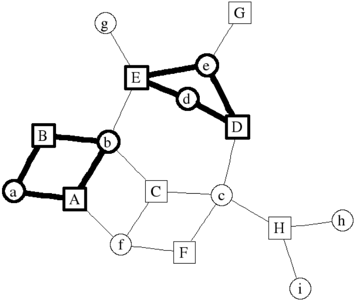
Example of a **non** simple-loop

Loop Characterization

Three categories of loops: simple, branching, and complex

Generalized loop: Any subgraph C of $\mathcal{G}$ such that each node in C has degree two or larger

branching-loop A Generalized Loop with more than one connected component



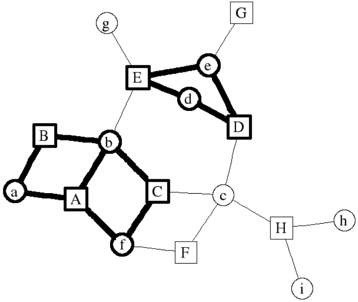
Example of a branching-loop composed of two simple-loops

Loop Characterization

Three categories of loops: simple, branching, and complex

Generalized loop: Any subgraph C of $\mathcal{G}$ such that each node in C has degree two or larger

branching-loop A Generalized Loop with more than one connected component



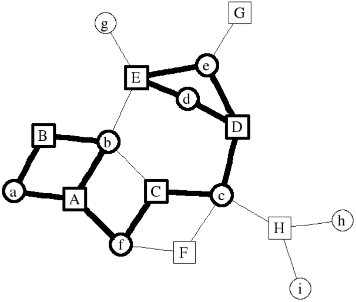
Example of a branching-loop composed of one non simple-loop

Loop Characterization

Three categories of loops: simple, branching, and complex

Generalized loop: Any subgraph C of $\mathcal{G}$ such that each node in C has degree two or larger

complex-loop A Generalized Loop which cannot be expressed as the **union** of two or more *different* simple-loops



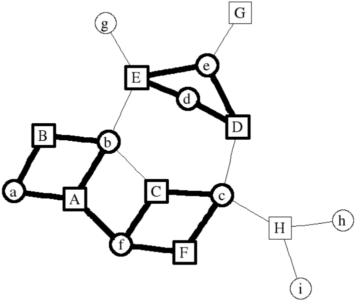
Example of a non branching complex-loop

Loop Characterization

Three categories of loops: simple, branching, and complex

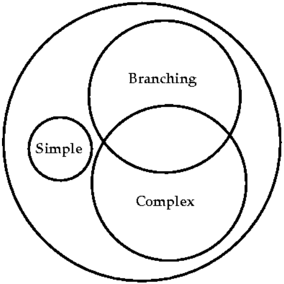
Generalized loop: Any subgraph C of $\mathcal{G}$ such that each node in C has degree two or larger

complex-loop A Generalized Loop which cannot be expressed as the **union** of two or more *different* simple-loops

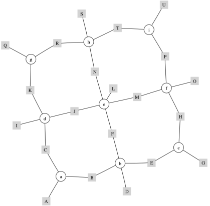


Example of a branching complex-loop

Generic diagram



- A small example: Ising 3x3



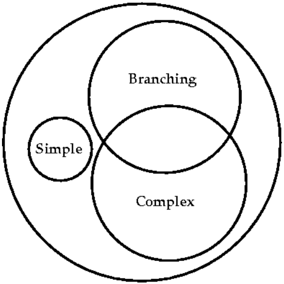
How many generalized loops?

The answer is . . .

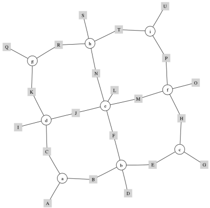
Loop Characterization

Summary

Generic diagram



- A small example: Ising 3x3



How many generalized loops?

The answer is . . . 42 !!!

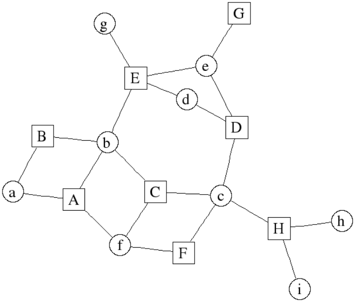
Outline

1. Introduction
   - Statistical Inference
   - Loop Calculus
2. Truncating Loop Expansion for BP
   - Loop Characterization
   - The TLSBP algorithm
   - Experimental Results
     - Ising Model
     - Medical Diagnosis
3. Conclusions

The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

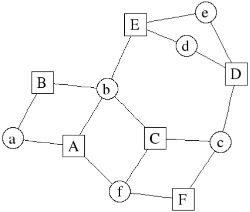
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

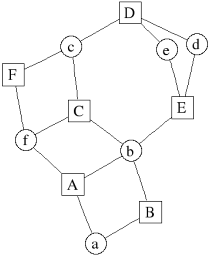
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

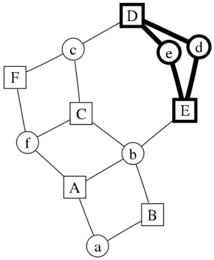
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

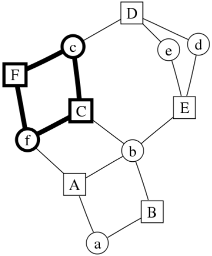
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

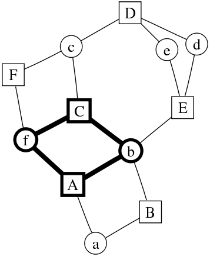
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

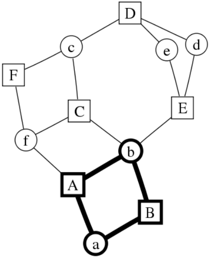
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

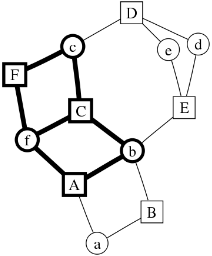
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

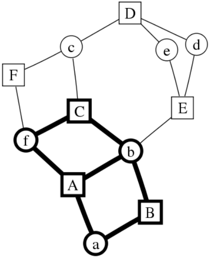
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

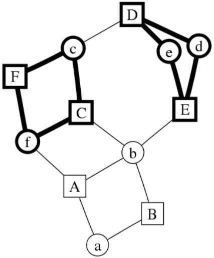
1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found

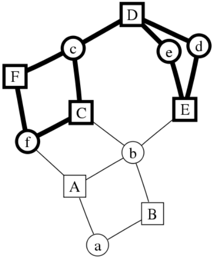


If branching → **Merge** must account for complex loops!

The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



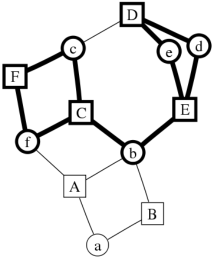
If branching → **Merge** must account for complex loops!

DFS bounded by M

The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



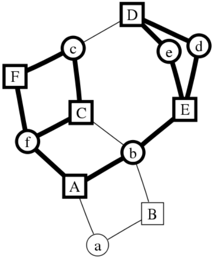
If branching → **Merge** must account for complex loops!

DFS bounded by M

The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ loops

1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



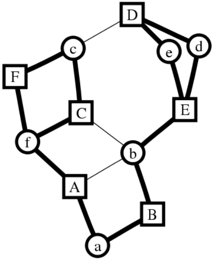
If branching → **Merge** must account for complex loops!

DFS bounded by M

The TLSBP algorithm

Arguments: (BP solution, S, M) returns $\mathcal{C}'$ **loops**

1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found



If branching → **Merge** must account for complex loops!

DFS bounded by M

The TLSBP algorithm

Arguments: (BP solution, **S**, **M**) **returns** $\mathcal{C}'$ loops

1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found

Given a subset $\mathcal{C}'$ of generalized loops:

- Compute approximation: $Z_{TLSBP} = Z_{\text{BP}}\left(1 + \sum_{C \in \mathcal{C}'} r(C)\right)$
- Compute marginals: **Clamping method**
  1. Run BP with variable i fixed to a value x_i
  2. Compute $Z^{x_i}_{TLSBP}$

$$b^{TLSBP}(x_i) = \frac{Z^{x_i}_{TLSBP}}{\sum_{x'_i} Z^{x'_i}_{TLSBP}}$$

The TLSBP algorithm

Arguments: (BP solution, **S**, **M**) **returns** $\mathcal{C}'$ loops

1. **Preprocessing**: remove 1-degree vertices recursively (2-core)
2. Find **S** simple-loops *generator set*
3. **Merge** pairs of loops iteratively until no new loops found

Characteristics

- A **complete** algorithm when parameters are relaxed
- Performs **Blind** search
- Computational Cost:
  - Loop search : instance dependent
  - Marginals : $\mathcal{O}(n^2 \cdot BP)$

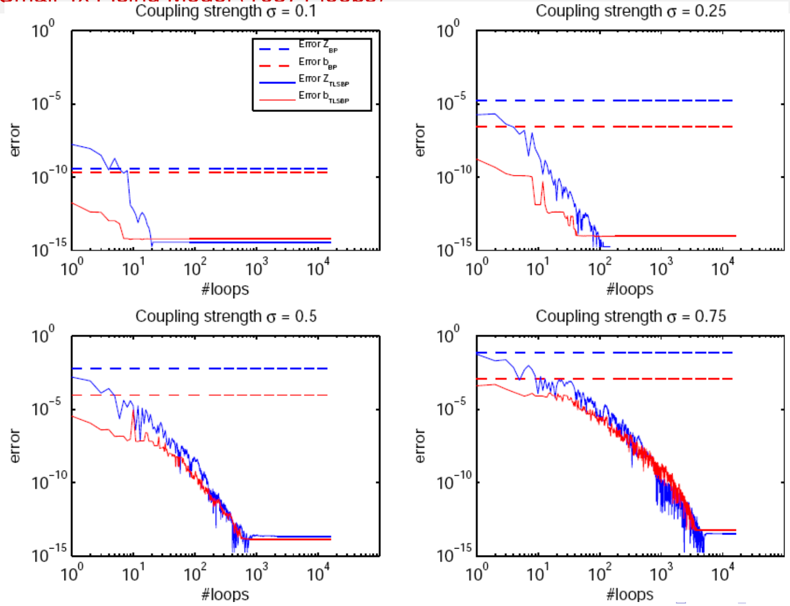
Outline

Setup

- Focus on a small ising grid 4x4
- Exhaustive enumeration is possible
- Sort all loops by decreasing contribution $|r(C)|$
- Study how error decreases as more loops are considered:
  - Error $Z_{TLSBP} = |\log Z - \log Z_{TLSBP}|$
  - Error $b_{TLSBP} = \max_i\{\max_{x_i}\{|b_i^{TLSBP}(x_i) - P(x_i)|\}\}$
- spin-glass configuration where:
  - single-node potentials $f_i(x_i) = exp(\theta_i x_i) \quad \theta_i \sim \mathcal{N}(0, 0.05)$
  - pairwise interactions $f_{ij}(x_i, x_j) = exp(\theta_{ij} x_i x_j) \quad \theta_{ij} \sim \mathcal{N}(0, \sigma)$

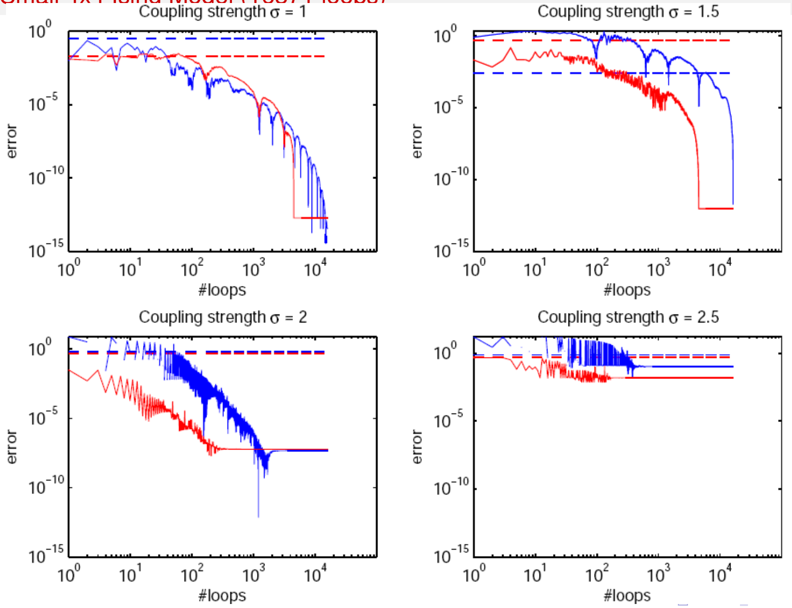
Experimental Results

Small 4x4 Ising Model (16371 loops)



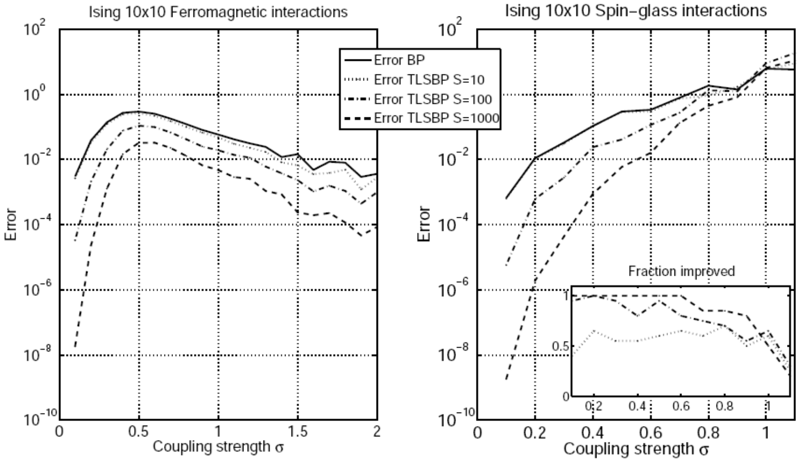
Experimental Results

Small 4x4 Ising Model (16371 loops)



Experimental Results

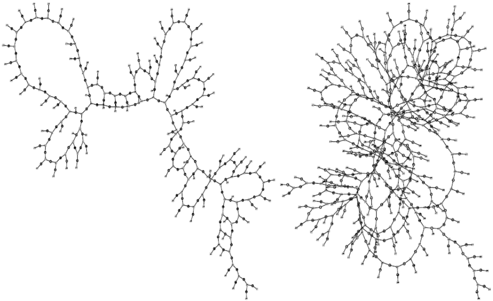
Ising Model 10x10 (coupling strength) Averages over 50 random instances (M=3)



Medical Diagnosis

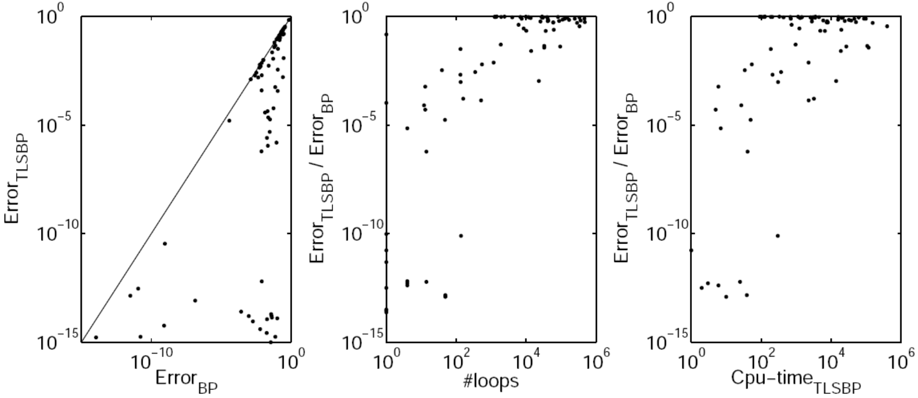
PROMEDAS

- 2-layered Bayesian Network:
  - Diagnosis layer: ~ 2000 diseases
  - Symptoms layer: ~ 1000 findings
- Each patient case results in a network
- Complexity depends on the set of findings



Medical Diagnosis

Results for 146 patient cases ($S = 50, M = 10$)



Summary

Conclusions

- TLSBP often improves upon the accuracy of the BP solution
- Truncating the series strongly depends on the coupling strength

Related work

- Truncation proposed in Chertkov [Chertkov M. & Chernyak V. '06b]
- Other Loop Corrections approaches: [Mooij et al. '07]

Current work

- Heuristic/Informed search for loops
- GBP comparison
- Apply Linear Response to compute marginals

THANK YOU!

References I/III

M. Chertkov and V.Y. Chernyak.
Loop series for discrete statistical models on graphs.
Journal of Statistical Mechanics, page P06009, 2006.

R.G. Gallager.
Low-density parity check codes.
MIT Press, 1963.

Kschischang, Frey, and Loeliger.
Factor Graphs and the Sum-Product Algorithm.
IEEETIT: IEEE Transactions on Information Theory, 47, 2001.

J. Pearl.
Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

References II/III

- J.S. Yedidia, W.T. Freeman, and Y. Weiss.
  Generalized belief propagation.
  NIPS (Proceedings of the 2000 Conference), 2001.
- Max Welling, Thomas Minka, and Yee Whye Teh.
  Structured region graphs: Morphing EP into GBP.
  Proceedings of the 21th UAI-05, page 609, Arlington, Virginia, 2005
- M. Chertkov and V.Y. Chernyak.
  Loop Calculus Helps to Improve Belief Propagation and Linear Programming Decodings of LDPC codes.
  Invited talk at 44th Allerton Conference, (September 27-29).
  URL `http://www.arxiv.org/abs/cs/0609154`.

References III/III

J.M. Mooij, B. Wemmenhove, H.J. Kappen, and T. Rizzo.
Loop corrected belief propagation.
In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.