Linear Programming Relaxations and Belief Propagation – An Empirical Study

CS295P

Linear Programming(LP) Relaxations

- Finding MAP in a graphical model
  - For an arbitrary graph: NP hard.
  - Need approximation.
- Linear Programming Relaxations
  - Used for approximating the MAP problem
  - guaranteed optimality

However,

  - Limited problem size
  - Requires powerful machines

Tree-Reweighted Belief Propogation

- TRBP
  - An alternative approximation
  - A variant of BP
- Advantages
  - Finds the same solution as LP relaxations
  - Applicable to large scale problems
  - Significantly faster than CPLEX

MAP

Conditional distribution: $$\Pr(x \mid y) = \frac{1}{Z} \prod_{<i,j>} \Psi_{ij}(x_i, x_j) \prod_i \Psi(x_i)$$

Define: $$\Psi_{ij}\left(x_i, x_j\right) = e^{-E_{ij}\left(x_i, x_j\right)}$$

So: $$x^* = \arg\min_x \sum_{<ij>} E_{ij}(x_i, x_j) + \sum_i E_i(x_i)$$

LP Relaxation

Define indicator function: $q_i(x_i), q_{ij}(x_i, x_j)$

LP relaxation of MAP:

minimize

$$J(\{q\}) = \sum_{<ij>} \sum_{x_i, x_j} q_{ij}(x_i, x_j) E_{ij}(x_i, x_j) + \sum_i \sum_{x_i} q_i(x_i) E_i(x_i)$$

subject to

$$0 \le q_{ij}(x_i, x_j) \le 1$$

$$\sum_{x_i, x_j} q_{ij}(x_i, x_j)$$

$$\sum_{x_i} q_{ij}(x_i, x_j) = q_j(x_j)$$

However..

The number of variables in LP relaxation:
Denote Ki the number of possible states of node i,
N_variables = sum(Ki) + sum(KiKj) for pairs of <i,j>

number of constraints:
N_constraints = sum(ki+kj+1) for pairs of <i,j>

For a 200*200 pixel image, each pixel takes on 30 values:
N_variables ~ 72million
N_constraints ~ 4 million

Tree-Reweighted BP

- Theory: [lecture by Prof. Sudderth]

- Number of iterations to converge:
  Not sure, but somehow faster than LP relaxation..
- Memory consumption:
  Implementation dependent

Benchmark Problems - Stereo Vision

http://vision.middlebury.edu/stereo/data/scenes2001/data/imagehtml/tsukuba.html

Goal: find the disparity of each pixel in a reference image. The disparity can be translated into depth from the camera.

Left

Right

Disparity

Benchmark Problems - Side-Chain Prediction and protein design

- Side-chain:

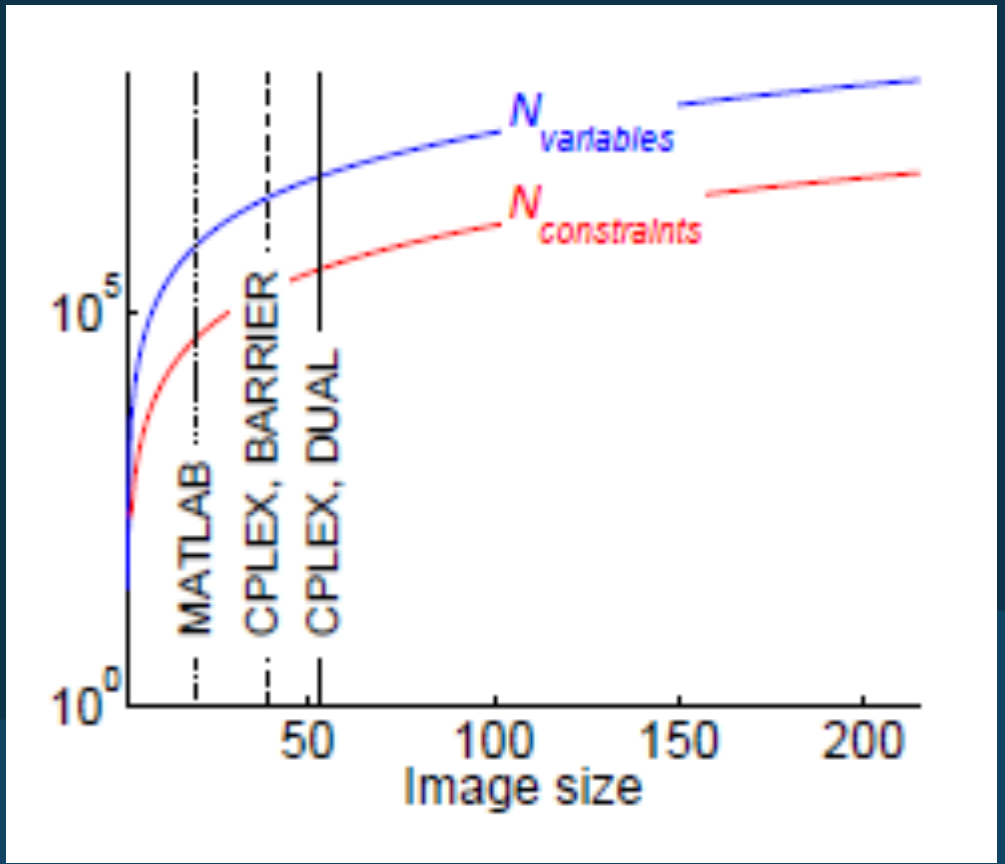
Proteins = chain(amino acid)
amino acid = Carbon base + NH_2 + COOH + side-chain
side-chain is bounded to the carbon base, etc..

Anyway:

- Important in protein folding and protein design
- Prediction is NP-complete

CPLEX vs TRBP in Stereo Vision

CPLEX: 50*50
TRBP: 250*250

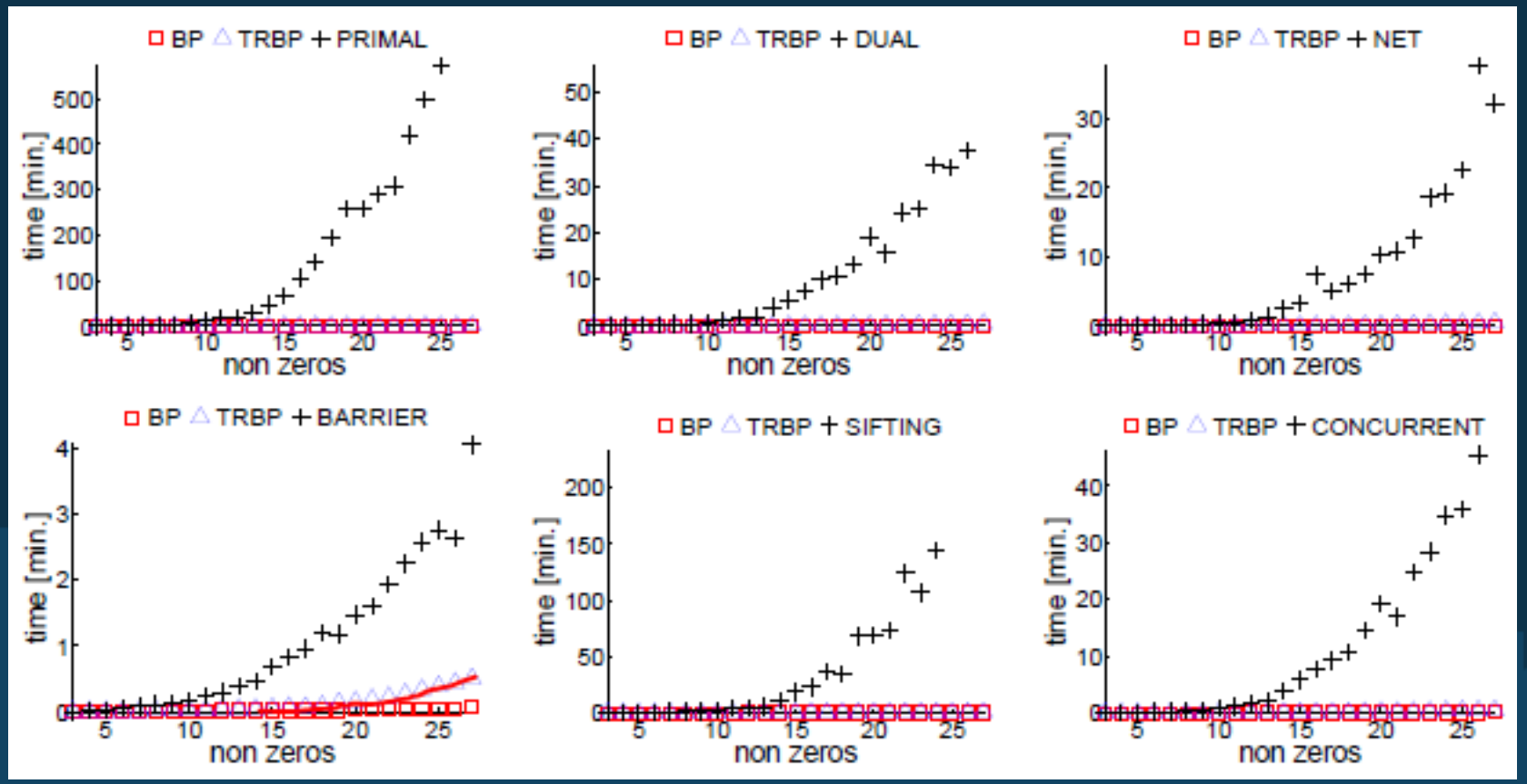


CPLEX vs TRBP in side-chain prediction and protein design

- For side-chain prediction, both CPLEX and TRBP can solve the LP relaxation for all proteins
- For protein design:
  - CPLEX: 3/97
  - TRBP: all

Runtime

Data set: a set of subproblems from the stereo benchmark set.



Summary

- TRBP is faster and memory efficient. -- Why?

The authors are not sure.. ("empirical study")
TRBP is not a general purpose LP solver. It can only solve a tiny fraction of linear programs with a very special structure.
Assumption:
LP Problem: minimize [cT]q subject to Aq=b and Cq<d
TRBP explicitly represents the graph, while CPLEX solvers explicitly represent A.