

2015 PGM Term Project: Belief Propagation for Combinatorial Optimization

Yongkyu Cho

Large-scale Stochastic Systems Lab.

December 3, 2015

Outline

- 1 Introduction
- 2 Shortest Path Problem
- 3 Implementation and Experiment
- 4 Conclusion

Outline

- 1 Introduction
 - Motivation
 - Objective
- 2 Shortest Path Problem
- 3 Implementation and Experiment
- 4 Conclusion

Motivation

Why BP for Optimization?

Research in Large-scale Optimization

- Mathematicians
 - Design distributed & Parallel algorithms
 - Pursuing mathematical beauty (sometimes unrealistic)
- Engineers
 - Implement them in large-scale systems
 - Pursuing realities

Why Belief Propagation?

- BP is powerful heuristic algorithm for solving computational inference problems in probabilistic graphical models.
- BP is proper to be coded in distributed manner in various systems.

Motivation

Can BP solve Combinatorial Optimization?

Consider the following GM and LP.

$$\text{for } x = [x_i] \in \{0, 1\}^n \text{ and } w = [w_i] \in \mathbb{R}^n,$$

$$\Pr[X = x] \propto \prod_i e^{-w_i x_i} \prod_{\alpha \in F} \psi_{\alpha}(x_{\alpha}),$$

Graphical Model

$$\begin{aligned} &\text{minimize} && w \cdot x \\ &\text{subject to} && \psi_{\alpha}(x_{\alpha}) = 1, \quad \forall \alpha \in F \\ &&& x = [x_i] \in [0, 1]^n. \end{aligned}$$

Linear Programming

$$\psi_{\alpha}(\mathbf{x}_{\alpha}) = \begin{cases} 1 & \text{if } A_{\alpha} \mathbf{x}_{\alpha} \geq b_{\alpha}, C_{\alpha} \mathbf{x}_{\alpha} = d_{\alpha} \\ 0 & \text{otherwise.} \end{cases}$$

If LP has an integral solution, that solution is equivalent to MAP.

Motivation

Can BP solve Combinatorial Optimization?

Sometimes, BP can solve Combinatorial Optimization problem.

Theorem (Park and Shin, 2015)

BP converges to the solution of LP if

- ① *LP has a unique and integral solution*
- ② *Each variable is associated to at most two factors*
- ③ *For every factor ψ_α , every $x_\alpha \in \{0, 1\}^{|\alpha|}$ with $\psi_\alpha(x_\alpha) = 1$, and every $i \in \alpha$ with $x_i \neq x_i^*$, there exists $\gamma \subset \alpha$ such that*

$$|\{j \in \{i\} \cup \gamma : |F_j| = 2\}| \leq 2$$

$$\psi_\alpha(x'_\alpha) = 1, \quad \text{where } x'_k = \begin{cases} x_k & \text{if } k \notin \{i\} \cup \gamma \\ x_k^* & \text{otherwise} \end{cases}.$$

$$\psi_\alpha(x''_\alpha) = 1, \quad \text{where } x''_k = \begin{cases} x_k & \text{if } k \in \{i\} \cup \gamma \\ x_k^* & \text{otherwise} \end{cases}.$$

Shortest Path problem is one popular example that satisfying the conditions above.

Objective of the term project

- Understanding BP for combinatorial optimization by implementing it for the *Shortest Path problem*.
- Compare the results for LP and BP (computation time, solution quality).

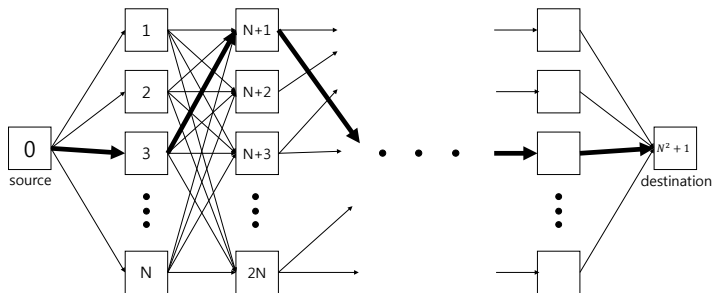
Outline

- 1 Introduction
- 2 Shortest Path Problem
 - Description
 - LP formulation
 - BP algorithm
- 3 Implementation and Experiment
- 4 Conclusion

Shortest Path Problem

Description

Given directed graph $G = (V, E)$ and non-negative edge weights $w = [w_e : e \in E] \in \mathbb{R}_+^{|E|}$, the *shortest path problem* is to find the shortest path from the source s to the destination t : it minimizes the sum of edge weights along the path.



Shortest Path Problem

LP formulation

minimize $w \cdot x$

subject to

$$\sum_{e \in \delta^o(v)} x_e - \sum_{e \in \delta^i(v)} x_e = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in V$$

$x = [x_e] \in [0, 1]^{|E|}$.

Annotations:

- $\delta^o(v)$: outgoing edges of vertex v
- $\delta^i(v)$: incoming edges of vertex v
- s : source node
- t : destination node

Shortest Path Problem

GM for Shortest Path Problem

$$\Pr[X = x] \propto \prod_{e \in E} e^{-w_e x_e} \prod_{v \in V} \psi_v(x_{\delta(v)}),$$

where the factor function ψ_v is defined as

$$\psi_v(x_{\delta(v)}) = \begin{cases} 1 & \text{if } \sum_{e \in \delta^o(v)} x_e - \sum_{e \in \delta^i(v)} x_e \\ & = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

Shortest Path Problem

Min-Sum BP Algorithm

Min-Sum Algorithm:

- 1) Initialize all messages to 0.
- 2) For e incident to v ,

$$m_{v \rightarrow e}^n(x) = \min_{x_{\partial v} : x_e = x} -\log \psi_v(x_{\partial v}) + \sum_{e' : e' \in \partial v - \{e\}} m_{e' \rightarrow v}^{n-1}(x_{e'})$$

- 3) For e incident to v and u ,

$$m_{e \rightarrow v}^n(x) = \log \phi_e(x) + m_{u \rightarrow e}^{n-1}(x)$$

- 4) Compute the beliefs at step n :

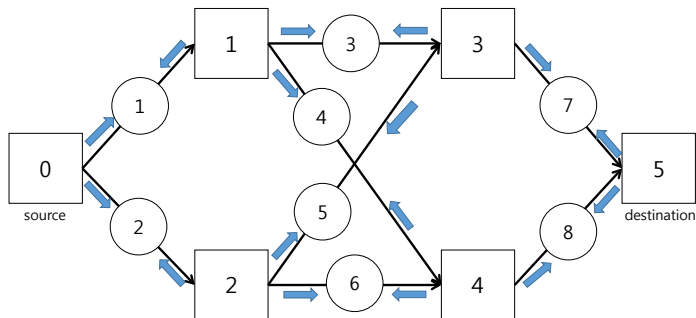
$$b_e^n(x) = \phi_e(x) + \sum_{v \in \partial e} m_{v \rightarrow e}^n(x)$$

- 5) Estimate membership of $e = (u, v)$ in the min path as

$$\overline{x_e}^n = \begin{cases} 1 & \text{if } b_e^n(1) < b_e^n(0) \\ 0 & \text{if } b_e^n(0) < b_e^n(1) \\ ? & \text{otherwise} \end{cases}$$

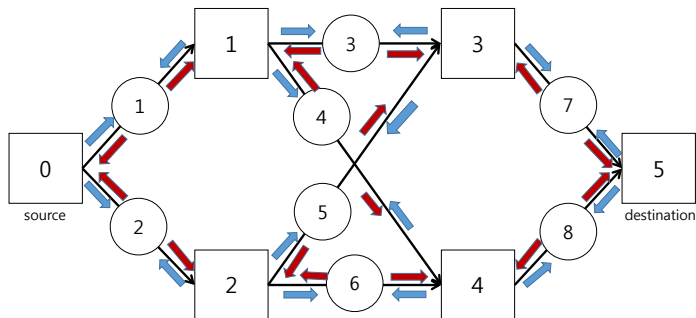
Shortest Path Problem

Protocol: Factors to variables



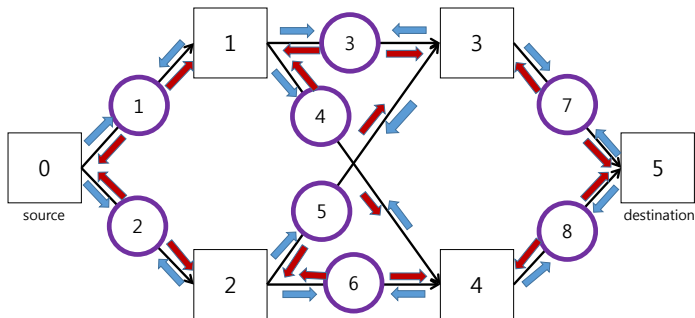
Shortest Path Problem

Protocol: Variables to Factors



Shortest Path Problem

Protocol: Computing beliefs



Outline

- 1 Introduction
- 2 Shortest Path Problem
- 3 Implementation and Experiment**
 - Experimental Environment
 - LP Result
 - BP Result (fail)
 - Result from Shin (2015)
- 4 Conclusion

Experimental Environment

Equipment	Intel Core i7-3770 3.40GHz, 8GB RAM, 64bit windows
LP Solver	CPLEX 12.6 with C++ Concert Technology
Language	Visual C++ 2010
Graph Instance	Randomly Generated (Edge weights \sim Unif(50,100))

LP Result

Number of vertices: $27(= 5^2 + 2)$

```

C:\Windows\system32\cmd.exe

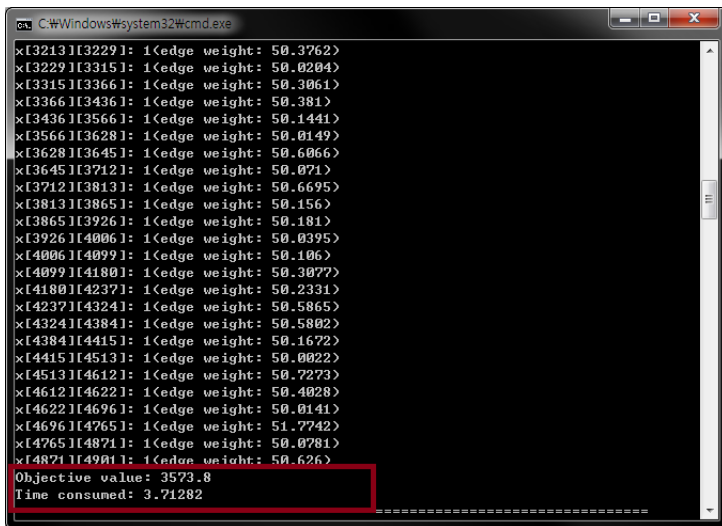
Shortest Path <Randomly generated DiGraph>
=====
We are generating a Random Directed Graph: Number of Vertices in a column?
5
A graph is generated!
=====
Which method do you use for getting the shortest path?
1-1: LP by CPLEX Solver
1-2: Message passing BP heuristic
1-3: EXIT
=====
1
Found incumbent of value 455.269305 after 0.00 sec. <0.01 ticks>
Tried aggregator 4 times.
MIP Presolve eliminated 2 rows and 86 columns.
Aggregator did 25 substitutions.
All rows and columns eliminated.
Presolve time = 0.00 sec. <0.24 ticks>

Root node processing <before b&c>:
  Real time           = 0.02 sec. <0.25 ticks>
Parallel b&c, 8 threads:
  Real time           = 0.00 sec. <0.00 ticks>
  Sync time <average>  = 0.00 sec.
  Wait time <average>  = 0.00 sec.
=====
Total <root+branch&cut> = 0.02 sec. <0.25 ticks>
x[0][4]: 1<edge weight: 68.0892>
x[4][9]: 1<edge weight: 53.7427>
x[9][12]: 1<edge weight: 50.3128>
x[12][16]: 1<edge weight: 51.4065>
x[16][24]: 1<edge weight: 52.1854>
x[24][26]: 1<edge weight: 55.8314>
Objective value: 331.568
Time consumed: 0
=====

```

LP Result

Number of vertices: $4902(= 70^2 + 2)$

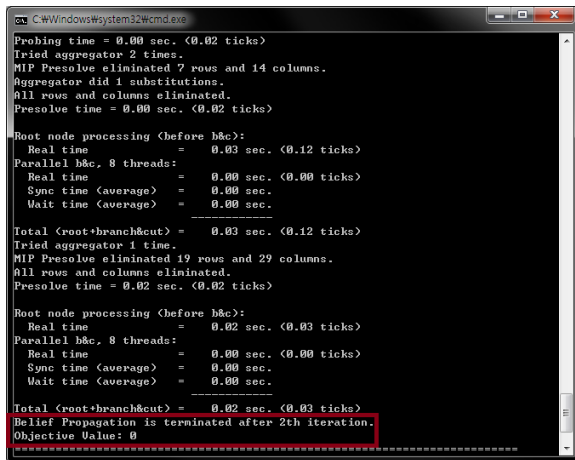
A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays a list of 28 edges, each with its endpoints and weight. The edges are listed in a columnar format. At the bottom of the list, the "Objective value: 3573.8" and "Time consumed: 3.71282" are displayed. A red rectangular box highlights the last two lines of the output.

```
C:\Windows\system32\cmd.exe
x[3213][3229]: 1<edge weight: 50.3762>
x[3229][3315]: 1<edge weight: 50.0204>
x[3315][3366]: 1<edge weight: 50.3061>
x[3366][3436]: 1<edge weight: 50.381>
x[3436][3566]: 1<edge weight: 50.1441>
x[3566][3628]: 1<edge weight: 50.0149>
x[3628][3645]: 1<edge weight: 50.6066>
x[3645][3712]: 1<edge weight: 50.071>
x[3712][3813]: 1<edge weight: 50.6695>
x[3813][3865]: 1<edge weight: 50.156>
x[3865][3926]: 1<edge weight: 50.181>
x[3926][4006]: 1<edge weight: 50.0395>
x[4006][4099]: 1<edge weight: 50.106>
x[4099][4180]: 1<edge weight: 50.3077>
x[4180][4237]: 1<edge weight: 50.2331>
x[4237][4324]: 1<edge weight: 50.5865>
x[4324][4384]: 1<edge weight: 50.5802>
x[4384][4415]: 1<edge weight: 50.1672>
x[4415][4513]: 1<edge weight: 50.0022>
x[4513][4612]: 1<edge weight: 50.7273>
x[4612][4622]: 1<edge weight: 50.4028>
x[4622][4696]: 1<edge weight: 50.0141>
x[4696][4765]: 1<edge weight: 51.7742>
x[4765][4871]: 1<edge weight: 50.0781>
x[4871][4901]: 1<edge weight: 50.626>
Objective value: 3573.8
Time consumed: 3.71282
```

BP Result (fail)

Number of vertices: $27(= 5^2 + 2)$

fail to debug...(unexpected result)



```
C:\Windows\system32\cmd.exe
Probing time = 0.00 sec. <0.02 ticks>
Tried aggregator 2 times.
MIP Presolve eliminated 7 rows and 14 columns.
Aggregator did 1 substitutions.
All rows and columns eliminated.
Presolve time = 0.00 sec. <0.02 ticks>

Root node processing (before b&c):
  Real time       =    0.03 sec. <0.12 ticks>
Parallel b&c, 8 threads:
  Real time       =    0.00 sec. <0.00 ticks>
  Sync time (average) =    0.00 sec.
  Wait time (average) =    0.00 sec.

Total (root+branch&cut) =    0.03 sec. <0.12 ticks>
Tried aggregator 1 time.
MIP Presolve eliminated 19 rows and 29 columns.
All rows and columns eliminated.
Presolve time = 0.02 sec. <0.02 ticks>

Root node processing (before b&c):
  Real time       =    0.02 sec. <0.03 ticks>
Parallel b&c, 8 threads:
  Real time       =    0.00 sec. <0.00 ticks>
  Sync time (average) =    0.00 sec.
  Wait time (average) =    0.00 sec.

Total (root+branch&cut) =    0.02 sec. <0.03 ticks>
Belief Propagation is terminated after 2th iteration.
Objective Value: 0
```

Result from Shin

IMA Workshop on GM, Statistical Inference, and Algorithms, 2015

- BP for *Maximum Weight Matching problem*
- Random bipartite graphs of 10,000 vertices and 5,000,000 edges
- BP provides 99.9+% approximation ratio while Gurobi and Blossom always output the optimal solution

Algorithm	Elapsed Time(sec)	Description
BP serial	2.08	Approximate BP algorithm
BP parallel	0.414	Accelerated BP with multi-threading and SIMD
Gurobi(LP solver)	8.67	Linear programming optimizer
Blossom	2.9	Augmented path algorithm

Conclusion

- BP can solve combinatorial optimization problem if some conditions are satisfied (Theorem. Park and Shin, 2015)
- BP solves combinatorial optimization faster than commercial LP solver attaining near-optimality.
- When BP is performed in distributed manner, the computation time can be much more reduced.

Thank you

References

- Sejun Park and Jinwoo Shin, *Max-Product Belief Propagation for Linear Programming: Applications to Combinatorial Optimization*, 2015.
- Jinwoo Shin, *Max-product Belief Propagation for Linear Programming in Combinatorial Optimization*, IMA Workshop on Graphical Models, Statistical Inference, and Algorithms, 2015.
- Nicholas Ruozzi and Sekhar Tatikonda, *s – t Paths Using the Min-Sum Algorithm*, In Communication, Control, and Computing, 2008 46th Annual Allerton Conference on, IEEE, 2008.