Outline:

1. What problem does it deal with: b-matching, 2-design?

2. Approach: formulate weighted b-matching as a prob distribution function on a graph.

3. Why BP: parallel, easy to run, approximates MAP.


\[ \text{Maximize:} \quad \sum_i x_i - \sum_{ij} A_{ij} \]

Consistency: Let \( M(U_i) = \{ U_i \} \), \( M(V_j) = \{ V_j \} \).

Denote \( x_i = M(U_i), y_j = M(V_j) \).

Then \( 2\epsilon(x_i, y_j) = |\text{XOR}(V_j \cup U_i) \cup (U_i \cup V_j)\).}

Now define MRF, s.t. MAP = \max b-match.

\[ P(x_i, y_j | x_i) = \exp \left( \frac{1}{2} A_{ij} \right) \]

But edge pot has nice structure.
Efficient/compact, exploit structure of $y_j$.

For a particular $y_j$ evaluate all $x_i$.

- If $x_i \in y_j$ then $\alpha_j(x_i, y_j) = 0$.
  - If $y_j \notin x_i$ sender isn't in receiver's matching list.
  - By $y_j$ (receiver's assignment).
- If $x_i \notin y_j$ for all $x_i$.
  - If $y_j \notin x_i$, then $\gamma_{ij}(x_i, y_j) = 0$.
  - Else if $\gamma_{ij}(x_i, y_j) = 1$.
  - $\mu_{i \rightarrow j}(y_j) = \max_{x_i : y_j \in x_i} \phi_i(x_i) \prod_{k : k \neq i} M_{v_k \rightarrow u_i}(x_i) \leftarrow$ independent of $y_j$.

Set to 1.

So the $y_j$ are divided into two categories, and we only need to calculate two REAL numbers. Normalize, so that $\forall y_j: u_i \in u_j$. $\mu_{i \rightarrow j}(y_j) = \frac{\prod_{k : k \neq i} M_{v_k \rightarrow u_i}(x_i)}{\prod_{k : k \neq i} M_{v_k \rightarrow u_i}(x_i)}$.

Sender isn't in receiver's matching list.

But what should $\mu_{i \rightarrow j}(y_j)$ be if $u_i$ by $y_j$?

Let's call the normalized message by $\bar{m}$. So $\bar{m}_{i \rightarrow j}(y_j) = 1$.

Now can actually drop $y_j$.

But keeps for clarity.

If $x_i \notin y_j$ (i.e. $u_i$ by $y_j$).

Otherwise, if connected then...
\[
\max_{X_i : V_j \in X_i} \sum_{k=k_j}^{i} \prod_{k} \tilde{m}_{V_k \rightarrow U_i}(X_i)
\]

Product over the rest b-1 elements in \( X_i \).

In numerator, are those which will be chosen.

1. \( V_j \in X_i \)
2. \( |X_i| = b \cdot \rho_i \) To maximize, the rest b-1 \( V_k \)'s will be the largest b-1 candidates of \( e_{Aik} \tilde{m}_{V_k \rightarrow U_i}(X_i) \) over k \( \neq j \)

so only need to find the bth largest \( e_{Aij} \tilde{m}_{V_j \rightarrow U_i}(X_i) \) over k \( \neq j \)

Finally, we can't efficiently reconstruct \( b_i(X_j) \), but can efficiently find its argmax.

\[
b_i(x_i) = \prod_{k \neq V_j \in X_i} e_{Aik} \tilde{m}_{V_k \rightarrow U_i}(X_i)
\]

Direct motivation? Like chinese restaurant process
Proof of Convergence

Assumption: 1. MAP is unique. Denoted by $M_{\bar{G}}$, given graph $G$, remember $M_{\bar{G}}$.

Let $E = W(M_{\bar{G}}) = \max_{M+M_{\bar{G}}} W(M) = \max W(M) - \frac{1}{2} \text{ convergence rate.}$

By uniqueness, we can decode at every node individually (separately). So we look at $M$, only i.e., $M_{\bar{G}}$ (i.e., $M_{\bar{G}}$ converges)

Unwrapped graph: (one of the standard methods to prove LBP convergence in special cases)

Example: Breadth-first search with revisiting (backtracking). Except first immediate backtracking. Example: $T$: always a tree (root node 0, fully-branching) on to many nodes and edges in $T$ have $G \rightarrow T$. Same local connectivity & potential functions as the corresponding node.

Show $P^{T}$ the original tree $T$, (root node 0, fully-branched)

Motivation: Want to simulate the $LBP$ on $G$ (with loops), by $BP$ on $T_d$ (without loop) with depth $d$.

Key observation: The belief at iteration $d$ of node $u_i$, during $LBP$ on $G$ (i.e., all the msg $u_i$ receives, thus far) is equivalent to the messages that $u_i$ receives in the unwrapped tree $T$ of depth $d$.

Show $P^{T}$, Ex 283 in PTT.

Given $T_d$. We can run max product on this tree, and find $M_{T_d}(r)$ for every node in $T_d$. We are interested in $M_{T_d}(r)$ due to simulation result $M_{T_d}(r) = \text{belief of } u_i \text{ after } LBP \text{ on } G \text{ for } d \text{ iterations}$.

W.T.S. $\exists do, s.t. \forall d > do$: $M_{T_d}(r) = \text{belief of } u_i \text{ after } LBP \text{ on } G \text{ for } d \text{ iterations}$.

Correct optimal match for $u_i$ in $G$.

Proof by contradiction. $\forall d > do$: $0$ weird limit. $\infty$ finite = infinite

Now we prove by construction: find such $do$. To do this, we need to check when $M_{T_d}(r) \text{ is possibly held, if it holds can hold only when } d < \text{ sufficiently large } d$.

We are done. Formally, W.T.S. $M_{T_d}(r) \Rightarrow d < \text{ sufficiently large } d$.

So equivalent to showing $d > do \Rightarrow M_{T_d}(r)$. So equivalent to show $d > do \Rightarrow M_{T_d}(r)$.

So now assume $M_{T_d}(r) \Rightarrow d \leq do$. Show that from above, transposes of last step.

We upper bound of from above, job is to find such a constant.

Basic idea: $M_{T_d}$ to $M_{T_d}$ outside. Then compared with original weight.
1. Map $M_G$ from $G$ to $T_d$. Since $G \rightarrow T$ is one-to-many, keep connection preserving $M_{T_d}$ on $T_d$.

2. Find an alternating path $P_T$ between $M_{T_d}(r)$ and $M_{T_d}(l)$.

   a. At root, pick one red & not blue, and one blue & not red.

   b. Possible because $M_G(e) + M_{T_d}(T)$.

   c. Degree constantly b. So red edge can find a blue & not red child.

left, right - go to root. We get a path $P_T$ on $T$ from one leaf to another leaf via the root, and its edges alternate between $M_{T_d}(b) \& M_{T_d}(r)$ (2d-1 hence)

Now you may imagine how the transplant $d$ is performed. Toggle the edges.

Now map $P_T$ back to $G$. Since $G$ only has 2n nodes, there must be cycles of at least $T = \frac{2d-1}{2n} > \frac{d-1}{n}$. With some remaining edges.

Show $P_T$ on cycle finder!

Suppose there are cycles $C = \{c_1, \ldots, c_t\}$

For $C_k$ toggle the edges, the new $M_G^{C_k}$ is still a b-matching, but no longer optimal.

\[
W(C_k \cap M_a) + W(M_A \setminus C_k) > W(M_a) + W(M_A \setminus C_k)
\]

\[
W(M_a) > W(M_a^{C_k}) + \alpha
\]

\[
W(C \cap M_A) - W(C \cap M_{T_d}) \geq \frac{d-1}{n} \alpha
\]

Finally, the remainder $P - P$ must be of even length. Wlog, $P$ starts with edge in $M_A$, ends with edge in $M_{T_d}$. To create a cycle $C_P$. Remove last edge $e_l$ from $M_{T_d}$ and replace it with an edge back to the first node $e_l$.

\[
W(C_P \cap M_a) - W(C_P \cap M_{T_d}) > \alpha
\]

Compensate the charge of edge $(e_l, e_l')$ with $(e_l, e_l')$. 

\[
W(e_l, e_l') > \alpha
\]
Experimental Results

1. Running time. Classical b-matching algo, such as balanced network flow used in GLOBE, has $O(bn^3)$ time. By $O(bn)$ to compute MSG for each node, $O(n)$ to cover 2 nodes and $O(n)$ iterations to converge. So overall $O(bn^3)$. But with much smaller constant factor in experiment.

Settings: Randomly generated bipartite graph with $n \in \{10,000\}$, $b \in [1, \frac{n}{2}]$, weights independently picked at random from uniform distribution in $[0,1]$. Code in Fix $b = \frac{n}{2}$, run 10
time.

medium time

2. For classification.

In kNN, some nodes may serve as hub nodes and labeling too many unknown examples while other training points are never used as neighbors. Using b-matching, each training point will contribute to the labeling of $b$ testing points only. (assuming $\#\text{train} = \#\text{test}$) downsample testing, run multiple folds.

Use negative Euclidean distance as $d_{ij}$

0. MNIST. 28x28 greyscale test 3, 5, 8, 4, 1, 9, 2, 0, 1. Cross-validate b & K in [1.5, 50].

Average accuracy over 20 random samplings.

Testing data are trained against various backgrounds (no longer iid) in training data, the background is just white.
Background replacement is like a translation of image vectors that preserves the general shape of distribution.

3. Textart: wood marble

Bum, torn by 10