Outline :   ①

1. What problem it deals with: $\boxed{\text{weighted b-matching}}$ bipartite, weights (not restricted to R^+). degree b (≥ 1). max. See Fig 1. edges uniform capacity

3. (2.F.) Approach. formulate weighted b-matching as a prob distribution function on GM, run BP (can be parallel) to find the MAP. Node state: Edge. why BP: parallel, early stop for approximate real-time

4. Significance ② theory: identifies another situation where BP on a loopy graphical model can provably and ① Practical: efficiently converge to the MAP (max-product) (single loop BP). still bn^3 but much faster

2. Application. ① combinatorial. Resource allocation: n supplier n customer, ship b supplies btwn customer supplier

② match bidders to sellers in auctions. ③ VLSI, Chinese Portman, shortest Path in undirected graphs with negative cost (no negative cycles)

④ In machine learning: modified KNN, pruning weighted affinity graph & removing noisy edges

Not first attempt: $b=1$ using BP, 2005; In a NIPS 2006 ~~describe~~ use GM for Combi Opt by blowing up state space exponentially, so standard BP is impractical.

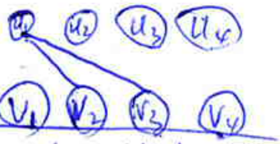
Algo.

Notation: $U = \{u_1 \cdots u_n\}$ $V = \{v_1 \cdots v_n\}$ $E = U \times V$ (fully). A_{ij} for edge (u_i, v_j) can be negative (asymm)

b-matching $M(u_i)$, $M(v_j)$ return set of neighboring ~~the~~ vertices in b-matching. In Fig 1.

e.g. $M(u_1) = \{v_1, v_2\}, \{v_1 v_3\} \cdots \{v_3, v_4\}$, 6 possibilities $\binom{n}{b}$ range. Pick one only!

So $\forall i \in \{1 \cdots n\}$. $|M(u_i)| = b$. $\forall j \in \{1 \cdots n\}$. $|M(v_j)| = b$



obj $\max_{M(u_i), M(v_j)} \sum_{i=1}^{n} \sum_{v_k \in M(u_i)} A_{ik} + \sum_{j=1}^{n} \sum_{u_k \in M(v_j)} A_{kj} \leftarrow W(M)$

Write out $M(u_i)$, $M(v_j)$, first and all candidate $M(u_i)$

Consistency: Eg. $M(u_1) = \{v_2, v_3\}$ $M(v_2) = \{u_2, u_3\}$ contradictory! need indicator

~~Denote $x_i = M(u_i)$, $y_j = M(v_j)$~~

① $v_j \bar{\in} M(u_i) \wedge u_i \bar{\in} M(v_j)$

① no link

② link

② $v_j \in M(u_i) \wedge u_i \in M(v_j)$

Denote $x_i = M(u_i)$. $y_j = M(v_j)$.

then $\psi_{ij}(x_i, y_j) = \overline{\text{XOR}}(v_j \in x_i, u_i \in y_j)$

$v_j \in \overbrace{M(u_i)}^{x_i}$	$u_i \in \overbrace{M(v_j)}^{y_j}$	
② T	T	✓
T	F	✗
F	T	✗
① F	F	✓
		$\overline{\text{XOR}}$

Now define MRF, s.t. MAP = max b-match, ~~and b-matching~~ still same graph, fully connected

but larger state space: $\binom{n}{b}$ x_i can be

$\phi_i(x_i) = \exp(\sum_{v_j \in x_i} A_{ij})$. $\phi_j(y_j) = \exp(\sum_{u_i \in y_j} A_{ij})$

LOOPY.

$P(\{x_i, y_i\}_{i=1}^{n}) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{j=1}^{n} \phi_j(y_j) \prod_{i,j=1}^{n} \psi_{ij}(x_i, y_j) \propto \exp(W(M))$

$\binom{n}{b}$ large space

consistency, not smoothness.

But edge pot has nice structure

BP: u_i x_i v_j y_j v_k $\quad m_{u_i \to v_j}(y_j) = \max_{x_i} \phi_i(x_i) \overbrace{\boxed{\psi_{ij}(x_i, y_j)}}^{\alpha_{ij}(x_i, y_j)} \prod_{k: k \neq j} m_{v_k \to u_i}(x_i)$

$k = 1 \cdots N$ since fully connected

Therefore diff y_j :

$b_i(x_i) = \phi_i(x_i) \prod_k m_{v_k \to u_i}(x_i)$

起草稿 ②

x_i, y_j is an assignment of u_i, v_j among the $\binom{n}{b}$ choices.

Efficient / compact. exploit structure of ψ_{ij}

motivation: $y_j \left(\right) = y_j \left(\psi_{ij} \right) \left(\right) \left(\right)$ (x_i, ...)

Same row in ψ_{ij}. then same msg

① y_j, y_j' both contain u_i then $\forall$ $\psi_{ij}(x_i, y_j) = \psi_{ij}(x_i, y_j')$ same row

For a particular y_j. ~~enumerate all x_i~~

~~If $u_i \in y_j$ then RHS of $\alpha_{ij}(x_i, y_j) = 0$ if~~

~~if $v_j \bar{\in} x_i$~~ sender is in receiver's matching list by y_j (receiver's assignment) ~~Sender & receiver~~

① If $u_i \in y_j$ (u_i, v_j connected) then only need to consider x_i which also says $u_i - v_j$ connect

for all x_i.

If $v_j \bar{\in} x_i$ disconnected then $\psi_{ij}(x_i, y_j) = 0$. ~~$\alpha_{ij}(x_i, y_j) = 0$~~.

Else if $\psi_{ij}(x_i, y_j) = 1$

$\therefore m_{u_i \to v_j}(y_j) = \max_{x_i : v_j \in x_i} \phi_i(x_i) \prod_{k: k \neq j} m_{v_k \to u_i}(x_i)$ ← Independent of y_j !

② If $u_i \bar{\in} y_j$

for all x_i

If $v_j \in x_i$. then $\psi_{ij}(x_i, y_j) = 0$.

Else if $v_j \bar{\in} x_i$. then $\psi_{ij} = 1$.

$\therefore m_{u_i \to v_j}(y_j) = \max_{x_i : v_j \bar{\in} x_i} \phi_i(x_i) \prod_{k: k \neq j} m_{v_k \to u_i}(x_i)$ set to 1

So the $\binom{n}{b}$ y_j are divided into two categories, and we only need to calculate two REAL numbers. Normalize, so that $\boxed{\forall y_j : u_i \bar{\in} u_j . \ m_{u_i \to v_j}(y_j) = 1}$

Furthermore

~~In ① $\prod_{k: k \neq j} m_{v_k \to u_i}(x_i) = \prod_{k: k \neq j, v_k \in x_i} m_{v_k \to u_i}(x_i) \prod_{k: k \neq j, v_k \bar{\in} x_i} m_{v_k \to u_i}(x_i)$~~

~~In ②~~

~~$m_{u_i \to v_j}(y_j)$~~

$\prod_{k \neq j, v_k \in x_i} m_{v_k \to u_i}(x_i)$

Sender isn't in receiver's matching ~~Sender & receiver are NOT connected~~

But what should $m_{u_i \to v_j}(y_j)$ be if $i \sim j$ by y_j ?

Let's call the normalized message by $\tilde{m}$. So $\tilde{m}_{u_i \to v_j}(y_j) = 1$

now can actually drop (y_j).

But keeps for clarity

if $u_i \bar{\in} y_j$ (i.e. u_i v_j ~~not connected~~

i.e. sender isn't in receiver's matching

Otherwise, if connected then.

$$\frac{m_{u_i\to v_j}(y_j)\;\; j\sim i}{m_{u_i\to v_j}(y_j)\;\; j\neq i} = \frac{\max_{x_i:v_j\in x_i} \phi_i(x_i)\prod_{k:k\neq j}\tilde m_{v_k\to u_i}(x_i)}{\max_{x_i:v_j\notin x_i}}$$

// $\tilde m_{u_i\to v_j}(y_j)$

$$= \frac{\max_{x_i:v_j\in x_i} \prod_{k:v_k\in x_i}\exp(A_{ik})\,\cancel{\phi_i(x_i)}\prod_{\substack{k:k\neq j\\ v_k\in x_i}}\tilde m_{v_k\to u_i}(x_i)\cdot\overbrace{\prod_{\substack{k:k\neq j\\ v_k\notin x_i}}\tilde m}^{1}}{\max_{x_i:v_j\notin x_i}\cancel{\phi_i(x_i)}\prod_{k:v_k\in x_i}e^{A_{ik}}\prod_{\substack{k:(k\neq j)\\ v_k\in x_i}}\tilde m_{v_k\to u_i}(x_i)\cdot\prod 1}$$

redundant

but $v_j\notin x_i$, $v_k\in x_i$ $\Big\}\Rightarrow k\neq j$

product over the rest b-1 elements in x_i

$$= \frac{\max_{x_i:v_j\in x_i} e^{A_{ij}}\prod_{\substack{k:k\neq j\\ v_k\in x_i}}e^{A_{ik}}\tilde m_{v_k\to u_i}(x_i)}{\max_{x_i:v_j\notin x_i}\prod_{k:v_k\in x_i}e^{A_{ik}}\tilde m_{v_k\to u_i}(x_i)}.$$

In numerator, ~~x_i to maximize~~ the feasible x_i ~~will be chosen among~~ are those which s.t. ① $v_j\in x_i$ ② $|x_i|=b$. To maximize, the rest b-1 v_k's will be the largest $\boxed{b-1}$ ~~candidates~~ values of $e^{A_{ik}}\tilde m_{v_k\to u_i}(x_i)$ over $k\neq j$

$$= \frac{e^{A_{ij}}\cdot\text{Product of } b-1 \text{ largest values of } e^{A_{ik}}\tilde m_{v_k\to u_i}(x_i) \text{ over } k\neq j}{\text{Product of } b \text{ largest values of } e^{A_{ik}}\tilde m_{v_k\to u_i}(x_i) \text{ over } k\neq j}$$

$$= \frac{e^{A_{ij}}}{\text{the } b^{th} \text{ largest } e^{A_{ik}}\tilde m_{v_k\to u_i}(x_i) \text{ over } k\neq j}$$

So only need to find the b^{th} largest: keep b largest $O(bn)$ each takes b.

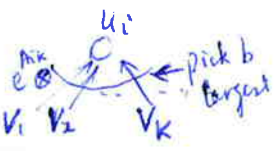
Finally, we can't efficiently reconstruct $b_i(x_i)$, but can efficiently find it's argmax

$$b_i(x_i) = \prod_{k:v_k\in x_i}e^{A_{ik}}\prod_{k:v_k\in x_i}\tilde m_{v_k\to u_i}(x_i)\prod_{k:v_k\notin x_i}\tilde m_{\to}(1)$$

$$= \prod_{k:v_k\in x_i}e^{A_{ik}}\tilde m_{v_k\to u_i}(x_i).$$ So to maximize $b_i(x_i)$ over x_i, just pick the b largest values of $e^{A_{ik}}\tilde m_{v_k\to u_i}(x_i)$ over $k=1\cdots N$

Nash Equili = ∃MRF s.t. MAP = best policy = LBP fixed point

msg:



direct motivation? Like chinese restaurant process

Proof of Convergence

Assumptions: 1. MAP is unique. ① denoted by M_G given graph G. remember M mean

② Let $\varepsilon = W(M_G) - \max_{M \neq M_G} W(M) = \max W(M) - \text{sec} \max W(M)$ $\frac{1}{\varepsilon}$ converge rate

By uniqueness, we can decode at every node individually (separately). So we look at u_1 only i.e. converge to $M(u_1)$ correct

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

T: ~~Dept~~ Breadth First search with revisiting (backtracking), except ~~but no~~ immediate backtracking. Example always a tree (First page ...

one to many
$G \to T$. nodes and edges in T have same local connectivity & potential functions as the corresponding node in G

show PPT the original tree T (root node u_1, fully branched)

$\boxed{\text{Motivation}}$ want ~~what~~ to simulate ~~mimic~~ the LBP on G (with loops) up to iteration d by BP on T_d (without loop) with depth d)

$\boxed{\text{Key observation}}$ The belief at iteration d of node u_1 during LBP on G (i.e. all the msg u_1 receives thus far) is equivalent to the ~~true~~ messages that r receives in the unwrapped tree T of depth d.

show Ex 2 & 3 in PPT.

Given T_d. we can run max product on this tree, and find $M_{T_d}(\cdot)$ for every node in T_d. We are interested in $M_{T_d}(r)$. Due to simulation result, $M_{T_d}(r)$ u_1 since $=$ belief of u_1 after ~~iteration~~ running LBP on G for d iterations. easy and ... we are inter...

W.T.S. $\exists d_0$, s.t. $\forall d > d_0$. $M_G(u_1) = M_{T_d}(r)$

for all sufficiently large d — correct optimal match for u_1 in G — MAP result of r on T_d

Proof by contradiction. $\forall d_0, \exists d > d_0 \neq$ ① weird limit. ① finite = infinite

Now we prove by construction: ~~find such d_0. To do this end, we need to check~~ analyze when ~~does $M_G(u_1) \neq M_{T_d}(r)$ can possibly hold. If it holds~~ inequality ~~can hold only when $d \leq$~~ const ~~sth~~ we are done. So ~~Formally, W.T.S. $M_G(u_1) \neq M_{T_d}(r) \Rightarrow d \leq sth$~~ const, then 作答

d_0 — can hold, can hold, can, must, must, ∞, d — can $\neq$, $=$

W.T.S. $d >$ some const $\Rightarrow M_G(u_1) = M_{T_d}(r)$

So equivalent to showing $M_G(u_1) \neq M_{T_d}(r) \Rightarrow d \leq$ some const ~~sth~~

So now assume $M_G(u_1) \neq M_{T_d}(r)$

we ~~upper~~ bounded d from above

our job is to find such a constant
show it ~~from above~~ — then compared with original weight by M_{T_d}

$kd - c$

Basic idea, ~~borrow~~ transplant the result of M_G to M_{T_d} it gives ~~better~~ weight ~~higher~~ reward start from r, linear to d, at the price of some constant offse

See illustrative example ~~$G, M_{G\to T_d}$~~ denote as $M^{G\to T_d}$

1. Map matching M_G from G to T_d. ✓ Since $G \to T$ is one-to-many ~~keeping~~ preserving local connectivi[ty]

show example PPT

M_{T_d}: MAP on T_d.

$M^{G\to T_d}_{T_d}$: M_G maped. To

2. Find an alternating path btw M_{T_d} & $M^{G}_{T_d}$

~~red~~ blue ~~blue~~ red

~~degree constantly b~~

① At root, pick one red & not blue, and one blue & not red.

Possible because $M_G(u_1) \neq M_{T_d}(r)$.

$\quad\quad\quad\quad\quad\quad \| \; M^{G}_{T_d}(r)$

② 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 $M^{G}_{T_d}(\cdot)$ & M_{T_d} (2d-1 hence)

Now you may imagine how the transplant is performed. toggle the edges

$V_2 U_4 V_1 U_1 V_3 U_4 V_1 \to V_2 U_4 V_1$ 看一[illegible] paper

Now map P_T back to G, since G only has $2n$ nodes, there must be cycles, ~~so at~~ ~~so at least~~ The longest cycle is $2n$.

cycle 画入图

加楼也

So at least $\tau \geq \frac{2d-1}{2n} \geq \frac{d-1}{n}$ with some remainder edges

$\leftarrow$ # edges in P_T

show PPT on cycle finding! linear! constant! pessimistic estimation

Suppose there are cycles $C = \{C_1 \cdots C_\tau\}$.

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

~~edges in C_k & M_G~~ ~~edges of M_G not in C_k~~ ~~edges in C_k & not in M_G but in M_{T_d}~~

$\cancel{W(C_k \cap M_G) + W(M_G \setminus C_k)} \quad \cancel{W(C_k \cap M_{T_d}) + W(M_G \setminus C_k)}$

$\| \; W(M_G) \quad\quad \| \; > W(M_G^{C_k}) + \varepsilon$

$\therefore \; W(C_k \cap M_G) - W(C_k \cap M_{T_d}) \geq \varepsilon$

$W(C \cap M_G) - W(C \cap M_{T_d}) \geq \tau\varepsilon \geq \frac{d-1}{n}\varepsilon$

Finally, the remainder P. P must be of even length. wlog. P starts with edge in M_G, ends with edge e_l 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_f. So

$\cancel{W(C_P \cap M_G) - W(C_P \cap M_{T_d}) \geq \varepsilon}$

~~Compensate for the charge of edge~~ $\cancel{((C_P \setminus e_l) \cap M_{T_d} \cup e_f) \geq \varepsilon}$

6

$W(P \cap M_G) - W(C_P \cap M_G)$

$\geq -\max_{e \in M_T} W(e) + \min_{e \in E} W(e)$

$W(C_P \setminus M_G) - W(P \setminus M_G) \geq 0$

$W(P \cap M_G) - W(P \cap M_{Td})$

~~$W(C_P$~~

$= W(P' \cap M_G) - W((P' \cap M_{Td}) \cup e_i)$

$= W((P' \cup e_f) \cap M_G)$

$P = C_P \cup e_i \setminus e_f$ 假设 $P = P' \cup e_i$

$e_i \in M_{Td}$

$\bar{\in} M_G$, $e_i \in M_{Td}$, e_f

$C_P = P' \cup e_f$

$e_i \in M_{Td}$

$P \setminus M_G = (C_P \cap e_f^c) \cup e_i$

$= (C_P \cup e_i) \cap e_f^c \cap M_G^c$ $\quad e_i \bar{\in} M_G$, $e_i \bar{\in} M$

$\supseteq (C_P \cup e_i) \cap M_G^c$

$\supseteq C_P \cap$

$= (C_P \cap e_f^c \cap M_G^c) \cup (e_i \cap e_f^c \cap M_G^c)$

$= (C_P \cap e_f^c \cap M_G^c) \cup (e_i \cap e_f^c)$ if $e_i \neq e_f$

$= (C_P \cap M_G^c \cup e_i) \cap e_f^c$

$= (C_P \cap e_f^c \cap M_G^c) \cup e_i$

P_G. P_T $e_i \in M_{Td}$ e_f. $G\ P_T\ e_f$ forms a cycle

$W(P_G) - W(P_T) - W(e_f) \geq \varepsilon$?

$\Rightarrow W(P_G) - W(P_T) - W(P_i)$

$= \underbrace{W(P_G) - W(P_T) - W(e_f)}_{\geq \varepsilon} + \underbrace{W(e_f)}_{\geq \min_{e \in E} W(e)} - \underbrace{W(P_i)}_{\geq -\max_{e \in W_T} W(e)}$ ②

For C_K. originally C_K^G, C_K^T are from G and M_{Td} resp. Now toggle the edges

Let $R_K^G = M_G \setminus C_K^G$. then $R_K^G \cup C_K^T$ is still a b-matching.

But $W(R_K^G \cup C_K^G) - W(R_K^G \cup C_K^T) \geq \varepsilon$. $\Rightarrow W(C_K^G) - W(C_K^T) \geq \varepsilon$.

$W(R_K^G \cup C_K^G) = W(R_K^G) + W(C_K^G)$, $\quad W(R_K^G \cup C_K^T) = W(R_K^G) + W(C_K^T)$

$\Rightarrow W\left(\sum_{k=1}^{\tau} C_K^G\right) - W\left(\sum_{k=1}^{\tau} C_K^T\right) \geq \tau\varepsilon \geq \frac{(d-1)\varepsilon}{N}$ ①

For remainder. 先讨论更

sum up ① ② $W(P_T \cap M_G) - W(P_T \cap M_{Td}) \geq \frac{d+n-1}{n}\varepsilon + \min_{e \in E} W(e) - \max_{e \in M_T} W$

So in tree T, replace $P_T \cap M_{Td}$ (blue, crossed edges) > 0 for $d \geq$ const $\left(\frac{1}{\varepsilon}n\right.$ $(\max - \min)\frac{n}{\varepsilon} - n +$

by $P_T \cap M_G$ (red, crossed edges). will give ~~higher~~ larger weight.

contradicts that M_{Td} is the max-weight tree.

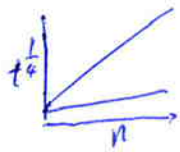
Experimental Results

1. Running time. Classical b-matching algo. such as balanced network flow used in GOBLI[N] ~~has~~ takes $O(bn^3)$ time. BP. $O(bn)$ to compute msg for each node, $O(n)$ ~~[illegible]~~ to cover $2n$ nodes and $O(n)$ iterations to converge. So overall $O(bn^3)$. But with much smaller const[ant] factor in experiment.

Settings. Randomly generated bipartite graphs with $n \in [10, 100]$, $b \in [1, \frac{n}{2}]$. weights independently picked at random from uniform distribution in $[0,1]$. Code in C

Fix $b=5$ cubic median time

$b = \lfloor \frac{n}{2} \rfloor$ quartic

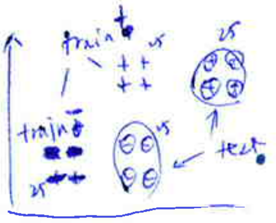


2. For classification.

In KNN, some nodes may serve as hub nodes and labeling too many unknown examples while other training points are never used as neighbors.

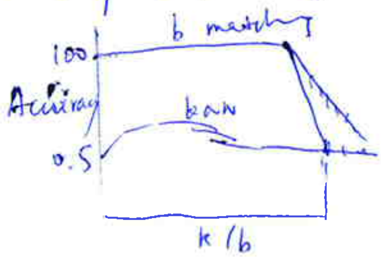
Using b-matching, each training point will contribute to the labelling of b testing points only. (assuming #train = #test) downsample testing, run multiple folds

Useful if test data is transformed in some way that preserves the shape of the distribution, but is translated or scaled to confuse KNN.



use negative Euclidean distance as A_{ij}

① Toy



② MNIST. 28x28 greyscale

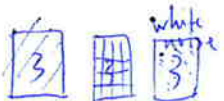
	3,*	5	8
trai	100	100	100
tes	100	100	100

cross validate b & k in [1, 300]

average accuracy over 20 random samplings

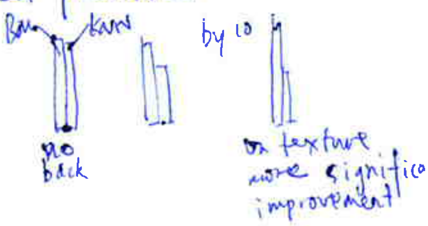
testing data are printed against various backgrounds (no longer iid)

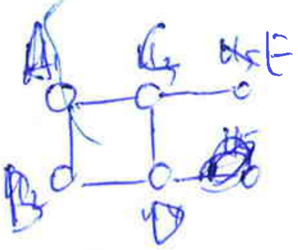
In training data, the background is just white

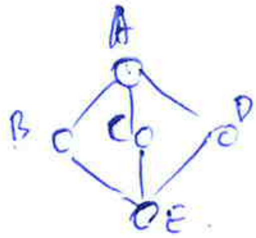


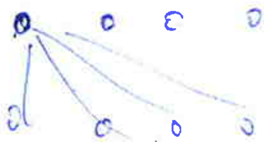
texture: wood marble

Background replacement is like a translation of image vectors that preserves the general shape of distribution.

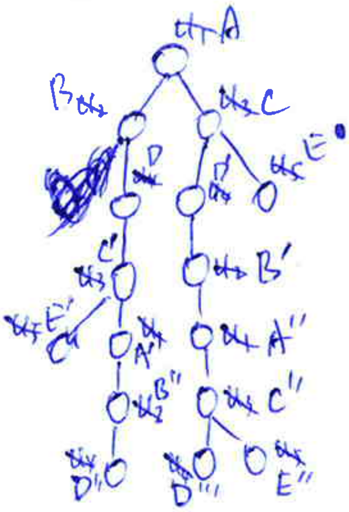


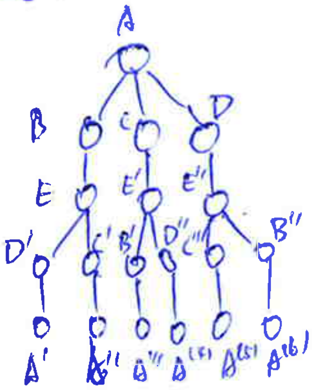






用黑线画等帐全连通

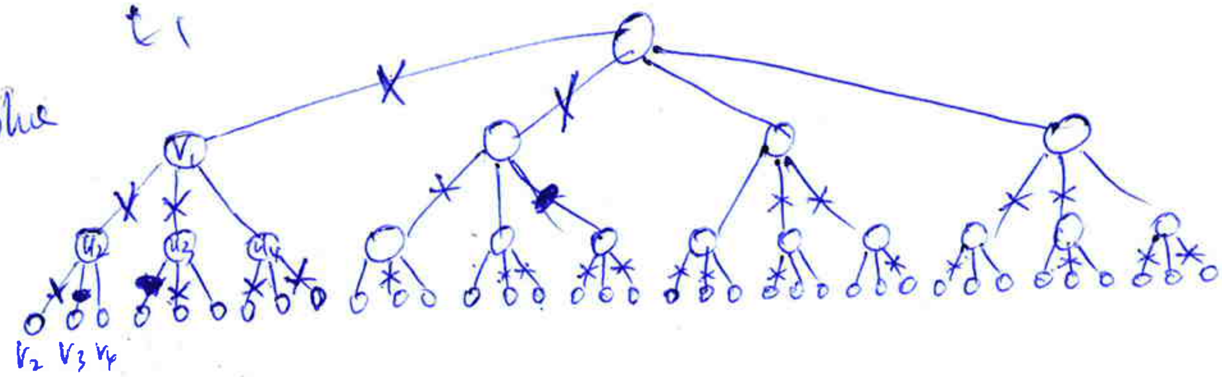




Ex 2

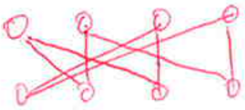
E3

E1

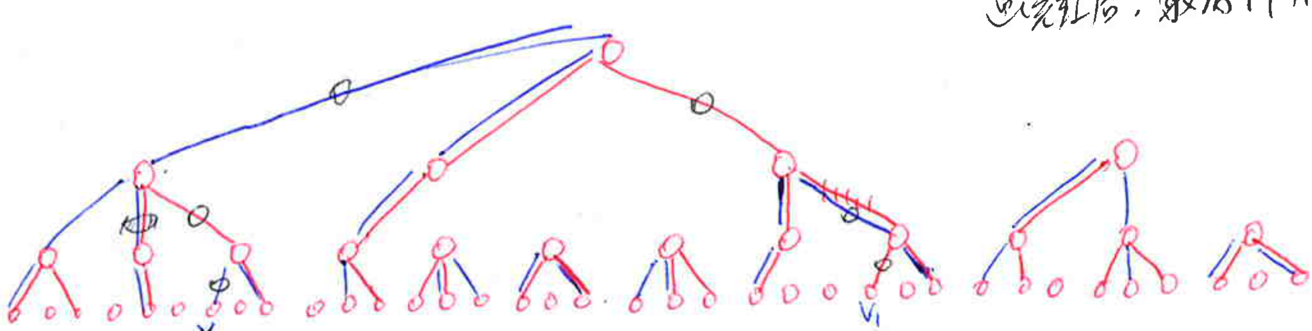


M_{Td} blue

M_G red M^G_{Td} from M_G mapped onto T. M_G is like Fig 1. M_G:

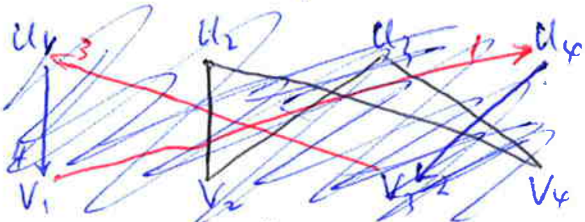


画完红后，最后P_T用X表示



$$\underbrace{V_2\ \overset{M_{Td}\ M^G_{Td}}{U_4}\ V_1\ U_1\ V_3\ U_4}_{C_1}\ V_1 \to \underbrace{V_2\ U_4\ V_1}_{P}$$

cycle:

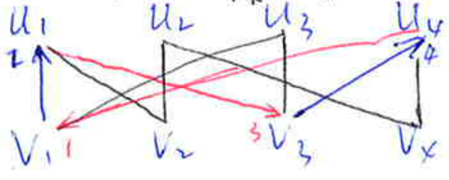


~~Black Blue:~~ cycle

Red: C^G_k from M^G_{Td} (from G)

Blue: C^T_k from M_{Td}

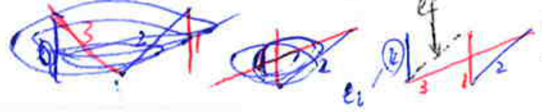
Black: $R^G_k = M_G \setminus C^G_k$



Red ∪ Black $= M_G$

Now replace red by blue

P:



Blue P_T

red P_G

Black e_f