Poznámky - Úvod do aproximačních a pravděpodobnostních algoritmů

Petr Chmel, ZS 2019/20

Average salary protocol

Required: if k people collude, they will not learn more than the sum of the rest.

Algorithm 1 (Average salary protocol)**.** P_i selects $R_{i,1}, \dots, R_{i,n-1} \in U[0, B)$ and $R_{i,n}$ so that $\sum R_{i,j} = S_i \mod nB$ where B is a known upper bound on the salaries.

Then, reveal $\sum R_{j,i}$.

Definitions

Definition 1 (Optimisation problem)**.** An optimisation problem is a quadruple (I, F, f, g), where

- I is the set of all instances
- F is a function which for $i \in I$ specifies the set of feasible solutions $F(i)$
- f is a function specifying the const of the feasible solution
- $g \in \{\min, \max\}$.

Definition 2 (NP-optimisation problem)**.** An NP-optimisation problem is an optimisation problem (I, F, f, g), where

- instances are finite strings from alphabet Σ
- for each instance and each solution, the size of the solution is polynomial in the size of the instance
- there exists a polynomial time decision procedure which tests whether $s \in F(i)$
- f is poly-time computable

Definition 3 (Optimal value, approximation ratio)**.** For an instance $I \in \mathcal{I}$, $\mathrm{OPT}(I)$ denotes the optimal value of a feasible solution.

Given an algorithm A for an optimisation problem, $A(I)$ denotes the value of a feasible solution found by A.

An algorithm A for an NP-optimisation problem (I, F, f, g) has an approximation ration R if

- A runs in polynomial time
- A always finds a feasible solution
- for every $i \in I$, $A(I) \leq R \cdot OPT(I)$ if minimising or $A(I) \geq OPT(I)/R$ if maximising

Travelling salesman problem

Input: $V = [n], d : V \times V \to \mathbb{R}_0^+$ a distance function
Output: $\pi : [n] \to [n]$ permutation such that $\sum_{i=1}^{n} d(\pi(i), \pi(i+1))$ is minimal

Theorem 1 (TSP nonapproximability)**.** Let $\alpha(n)$ be a polytime-computable function. Then, if $P \neq NP$, there is no $\alpha(n)$-approximation algorithm for TSP.

Definition 4 (Metric space)**.** A metric space is $(M, \delta), \delta : M \times M \to \mathbb{R}_0^+$ such that

1. $\forall x, y \in M : \delta(x, y) = 0 \Leftrightarrow x = y$
2. $\forall x, y \in M : \delta(x, y) = \delta(y, x)$

3. $\forall x, y, z \in M : \delta(x,y) + \delta(y,z) \geq \delta(x,z)$

Algorithm 2 (TSP-MST)**.** 1. Compute the MST of G

2. Find a Eulerian tour in the graph obtained from MST by doubling every edge

3. Obtain a Hamiltonian tour by reomving already visited vertices

Theorem 2 (TSP-MST is 2-apx)**.** TSP-MST is a 2-approximation algorithm

Důkaz. $cost(MST) \leq OPT(I)$
$A(I) \leq cost(tour) = 2cost(MST) \leq 2OPT$. ⊞

Algorithm 3 (CHRISTOFIDES)**.** 1. Compute MST

2. Find a min-cost perfect matching M on the vertices with odd degree in the MST

3. Find a Eulerian tour on MST and M

4. Get a Hamiltonian cycle by removing already visited vertices

Theorem 3 (CHRISTOFIDES is 1.5-apx)**.** CHRISTOFIDES algorithm is a $\frac{3}{2}$-approximation algorithm.

Důkaz. $A(I) = d(Hamtour) \leq d(Eultour) = cost(MST) + cost(M) \leq 1.5OPT$. ⊞

Probability review

Quicksort

Algorithm 4 (QS)**.** IN: n distinct numbers
QS(S):

1. if $|S| = 1$, return S;

2. choose uniformly at random a pivot p from S.

3. $S^- := \{x \in S : x < p\}, S^+ := \{x \in S : x > P\}$

4. Return $QS(S^-) + p + QS(S^+)$

Analysis: assume $y_1, \ldots, y_n$ are sorted. $X_{i,j} := 1[y_i, y_j$ are compared during the run$]$ for $i < j$, $Y_{i,j}$ the event itself. $\forall i, j : y_i, y_j$ compared at most once. Fix y_i, y_j. Event B_l : a pivot from $\{y_i, y_j\}$ is chosen in recursion level l for the first time.

$l \neq l'$: $B_l, B_{l'}$ are disjoint.

For every $\omega \in \Omega \exists l : \omega \in B_l \Leftrightarrow \bigcup_{l=1}^{n} = \Omega$

$P[Y_{i,j}|B_l] = \frac{2}{j-i+1}$, hence $P[Y_{i,j}] = \sum_{l=1}^{n} P[X_{i,j}|B_l] \cdot P[B_l] = \frac{2}{j-i+1} \sum P[B_l] = \frac{2}{j-i+1}$

Therefore $\sum\sum EX_{ij} = \sum\sum \frac{2}{j-i+1} = \sum_{d=1}^{n-1} \frac{2}{d+1} \cdot (n-d) = 2\sum_{d=1}^{n-1}(\frac{n}{d+1} - \frac{d}{d+1}) \geq 2nH_n \in \mathcal{O}(n \log n)$.

Contention resolution in a distributed system

Problem: n processors acess a single, shared database, synchronous, operate in discrete rounds without any direct communication, only one can access the database at a time.

Goal: design a protocol that would ensure access to the dabatase for each processor quickly.

Algorithm 5 (ACCESS)**.** In each round, try to access the database with probability $1 > p > 0$ (optimally, $1/n$).

Theorem 4 (ACCESS works whp in $2en \ln n$ rounds)**.** With probability at least $1 - \frac{1}{n}$, all processors succeed in accessing the database at least once within $t = 2en \ln n$ rounds for $p = 1/n$.

Důkaz. Choices are independent; event $A_{i,t} : P_i$ succeeds in round $t : P[A_{i,t}] = \frac{1}{n} \cdot (\frac{n-1}{n})^{n-1} \geq \frac{1}{en}$.

Event $F_{i,t}$ - P_i does not suceed in any of the first t rounds: $P[F_{i,t}] = \prod(1_P[A_{i,r}]) \leq (1 - \frac{1}{en})^t = [(1 - \frac{1}{en})^{en}]^{\frac{t}{en}} < e^{-\frac{t}{en}} = e^{-2\ln n} = n^{-2}$ ⊞

Global minimum cut

Input: $G = (V, E)$
Output: $S \subseteq V : \emptyset \neq S \neq V$
Goal: minimize $E(S, V \setminus S)$.

This can be solved in $(n-1)$ runs of Ford-Fulkerson or Dinic.

Want to it quicker randomly:

Observations: Every cut in G/e corresponds to a cut in G of the same size, if C is a cut in G and $e \notin C$, then there is a cut of size $|C|$ in G/e.

Algorithm 6 (RANDMINCUT)**.** while G has more than two vertices, choose an edge uniformly at random and then contract that edge (multiple edges allowed after contraction)

return the cut corresponding to the two remaining vertices

Theorem 5 (RANDMINCUT and global minimum)**.** The algorithm returns a global minimum with probability $\geq \frac{2}{n(n-1)}$.

Důkaz. If the algorithm never chooses and edge $e \in C$, where C is a minimum cut, then it finds C - fix such a C. In an n-vertex graph with mincut of size k, $\deg(v) \geq k \forall v \in V$, hence $kn/2 \leq |E|$. Probability that the algo picks an edge from C in the first iteration is $\leq 2/n = \frac{k}{kn/2}$. The number of vertices decreases by one in each iteration.

Lemma (without proof): $P[E_1 \cap \ldots \cap E_k] = \prod [E_i | \bigcap_{j=1}^{i-1} E_j]$

Define E_i : no edge from C is contracted in i-th iteration: $P[E_1] = \frac{n-2}{n}, P[E_i | \bigcap E_j] \geq 1 - \frac{k}{kn_i/2} = \frac{n_i - 2}{n_i} = \frac{n-i-1}{n-i+1}$.

Now $P[\bigcap E_i] \geq \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} = \frac{2}{n(n-1)}$. ⊞

Scheduling on identical machines

Problem: n jobs with lengths p_i, m identical machines. Output: partition of $[n]$ into m sets with the maximal sum of $p_i, i \in S_l$ minimised.

Algorithm 7 (LOCAL SEARCH for SCHEDULING)**.**

1. Start with any schedule
2. If there is a job which ends the latest and can be reassigned to an earlier starting time, do so and repeat.
3. Output the final schedule

Theorem 6 (LOCAL SEARCH is 2-apx)**.** LOCAL SEARCH is a 2-approximation algorithm.

Důkaz. OPT$\geq$ max p_j
OPT$\geq \frac{\sum p_j}{m}$
$c_{min} \leq \frac{\sum p_j}{m}$
$S_j \leq c_{min}$
$C_{max} = S_j + p_j \leq C_{min} + p_j \leq C_{min} + OPT \leq 2OPT$

(We can even use $S_j \leq \frac{\sum_{p_i - p_j}}{m}$) for $2 - 1/m$ approximation ratio. ⊞

Algorithm 8 (GREEDY SCHEDULING)**.**

1. Order the jobs arbitrarily
2. Process the jobs one by one and assing the job to the least loaded machine

Theorem 7 (GREEDY is also 2-1/m-apx)**.** GREEDY algorithm is also a $(2 - \frac{1}{m})$-approximation algorithm.

Důkaz. LOCAL cannot improve GREEDY. ⊞

Remark (Competitive ratio)**.** Competitive ratio is a ratio of an online algorithm when compared to the OPT of an offline precise algorithm.

Theorem 8 (GREEDY competitive ratio is $2 - 1/m$)**.** GREEDY algorithm also has a competitive ratio of $(2 - \frac{1}{m})$.

Algorithm 9 (LARGEST PROCESSING TIME)**.** GREEDY, except the order is from longest to shortest job

Theorem 9 (LPT is 4/3-apx)**.** LPT is a 4/3-approximation algorithm.

Důkaz. $p_n \leq OPT/3 : C_{MAX} = S_n + p_n \leq 4/3OPT$
$p_n > OPT/3$: at most two jobs are assigned to each machine in the optimal schedule. Then if $n \geq m + i$, then $OPT \geq p_{m-i+1} + p_{m+i}$ and one job of size $\geq p_{m-i+1}$ must be paired.

At the same time, at least $n - m$ jobs from the $p_1, \ldots, p_m$ have size $\leq 2/3OPT$. From this, we can get 4/3 easily, and even conclude that this is the optimum. ⊞

Bin packing

Input: $a_1, \ldots, a_n \in (0, 1)$
Output: partition on $[n]$ into some sets so that in every set, the values sum up to at most one.
Goal: minimise m

Algorithm 10 (FIRST/BEST/ANY FIT BIN PACKING)**.** For every item, let i be the first/the most loaded/some bin such that the item fits in the bin and place it there (if none, add a new bin).

Theorem 10 (ANYFIT is 2-apx)**.** Every ANYFIT greedy algorithm is a 2-approximation algorithm.

Důkaz. $B_l :=\sim_{i \in I_l} a_i$

$B_i + B_{i+1} > 1 \Rightarrow 2OPT \geq 2\sum B_i > M \Rightarrow OPT \geq \sum B_i \geq m/2$, which yields the 2-approximation ratio. ⊞

Edge disjoint paths in graphs

Input: $G = (V, E), k$ pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$
Output: $I \subseteq [k]$ and a path P_i for each $i \in I$ between s_i and t_i with all the paths edge disjoint.
Goal: maximise $|W|$.

Remark. The decision problem is NP-hard if k is part of the input
For fixed k, solvable in poly-time on undirected graphs, but NP-hard for $k = 2$ on directed graphs.

Algorithm 11 (GREEDY EDP)**.**

1. $I := \emptyset$
2. find a shortest path that connects some $s_i, t_i, i \notin I$ and is edge disjoint. If no such, goto 4
3. Add the path and goto 2
4. return I, P_i

Theorem 11 (GREEDY EDP is a $2\sqrt{m}+1$-apx)**.** GREEDY EDT is a $2\sqrt{m}+1$ approximation algorithm.

Důkaz. $OPT_l = \{i \in OPT : |P_i^*| > \sqrt{m}\}, OPT_s = \{i \in OPT : |P_i^*| \leq \sqrt{m}\}$

$|OPT_l| \leq \sqrt{m}$ (via double counting or simple logic). Take a P_i^* which is not in the result and is at most $\sqrt{m}$ long. Then there exists a path $P_j \in S$ in our result which shares an edge with it. One such path can only block a single path with one edge.

Hence every path in OPT is either long, or its terminals are also connected, or its blocked by a short path chosen by the algorithm. Therefore $|OPT| \leq \sqrt{m} + |I| + \sqrt{m}|I| \leq (2\sqrt{m}+1)|I|$. ⊞

Paths in graphs with capacities

Same as before, except every edge has a capacity

Algorithm 12 (GREEDY EDP capacities)**.** 1. $I := \emptyset, m := |E|, \beta := \lceil m^{1/(c+1)} \rceil, d(e) := 1 \forall e \in E$

2. find a shortest path P with respect to d across all $i \notin I$, if it does not exist or breaks the capacity rule, then goto 4
3. Add the path and for all $e \in P : d(e) := \beta \times d(e)$, goto 2
4. return

Theorem 12 (GREEDY is $\mathcal{O}(m^{1/(c+1)})$-apx)**.** GREEDY is $\mathcal{O}(m^{1/(c+1)})$-approximation algorithm.

Důkaz. Take an instance of $|I|$ and OPT. Consider a path short if its length is less than β^c with respect to current values of d. As long as we take short paths, we're ok as the paths cannot break the capacity limit.

From now on: we consider length d' from the end of the last iteration, when greedy chose a short path.

If OPT joins (s_i, t_i) and greedy does not, then $d'(P_i^*) \geq \beta^c$. There are at most $OPT - |I|$ such i's, and every edge from E is used by at most c paths P_i^*. Hence $d'(E) \geq \beta^c(OPT - |I|)/c$

Now for an upper bound: $d'(E) = m \leq \beta^{c+1}$ at the start of the algorithm. Then, by adding a path, we may at most multiply the edges by β, hence $d'(E) \leq (1 + |I|)\beta^{c+1}$.

Therefore $\beta^c(OPT - |I|)/c \leq d'(E) \leq (1 + |I|)\beta^{c+1} \Rightarrow OPT \leq (1 + |I|)c\beta + |I|$ and the approximation follows. ⊞

Maximum satisfiability

Input: n boolean variables, m clauses
Output: assignment of true/false to the variables
Goal: maximize the number of satisfiable clauses
Assumptions: no literal repeats in a clause and at most one of $x_i, \overline{x_i}$ appear in any clause

Algorithm 13 (RAND-SAT)**.** Randomly assign each variable true/false with probability 1/2 independently.

Theorem 13 (RAND-SAT is a 2-apx)**.** C_j : indicator variable Y_j =value of the clause. $P[Y_j = 0] \leq \frac{1}{2^{|C_j|}}$.

Then $\mathbb{E}[\sum Y_j] \geq m/2 \geq OPT/2$

Algorithm 14 (BIASED-SAT)**.** If x_i appears more frequently than $\neg x_i$, then $a_i = 1$ with probability $\phi - 1 = (\sqrt{5} - 1)/2$ and 0 otherwise. Otherwise $a_i = 0$ with probability $\phi - 1 = (\sqrt{5} - 1)/2$ and 1 otherwise.

Theorem 14 (BIASED-SAT is $\phi - 1$-apx)**.** BIASED-SAT is a $\phi - 1$-approximation algorithm.

Důkaz. $P[Y_j = 1] = 1 - p^\alpha(1 - p)^\beta \geq 1 - p^{\alpha+\beta} \geq 1 - p^2 = 1 - (\phi - 1)^2 = \phi - 1$ ⊞

Algorithm 15 (LP-SAT)**.** $f(C_j) := \sum_{i:x_i \in C_j} y_i + \sum_{i:\neg x_i \in C_j}(1 - y_i)$.

max $\sum z_j$ where $f(C_j) \leq z_j$, $0 \leq y_j, z_j \leq 1$

Construct the LP and find its optimal solution y^*, z^* and return $a_i = 1$ with probability y_i^*.

Theorem 15 (LP-Sat is a $1 - 1/e$-apx)**.** LP-SAT is a $1 - 1/e$ approximation algorithm.

Důkaz. $P[C_j(a) = 1] \geq 1 - (1 - z_j^*/k_j)^{k_j} \leq z_j^*(1 - (1 - 1/k_j)^{k_j}) \geq (1 - 1/e)z_j^*$ ⊞

Algorithm 16 (BEST-SAT)**.** W. probability 1/2 run LP-SAT, else run RAND-SAT.

Theorem 16 (BEST-SAT is a 3/4-apx)**.** BEST-SAT is a 3/4-approximation algorithm.

Důkaz. $P[C_j(a) = 1]$ based on the length of the clause: $1, 2, 3+$. ⊞

Vertex and set cover

Input: m subsets $S_i \subseteq [n]$, $c_1, \ldots, c_m \in \mathbb{R}_0^+$.
Output: $I \subseteq [m] : \bigcup_{i \in I} S_i = [n]$
Goal: min $\sum_{i \in I0} c_i$.

Two parameters: $f := \max_{e \in [n]} |\{i : e \in S_i\}| \leq m, g := \max_{i \in [m]} |S_i| \leq n$

Algorithm 17 (GREEDY-SC)**.** 1. $I := \emptyset, E := \emptyset$

2. while E does not cover the whole $[n]$: find the set with the smallest ratio $c_j/|S_j \setminus E|$, then $q_e := p_{j_0} forall e \in S_j \setminus E$ for j_0 the chosen index

3. return I

Theorem 17 (GREEDY-SC is $\ln g$-apx)**.** GREEDY-SC is $\ln g$-approximation algorithm.

Důkaz. $\sum q_e = \sum c_j = ALG$

Take S_j reordered so that e_k was covered first, e_1 last. We claim: $q_{e_i} \leq c_j/i$ - at least i elements of S_j were not covered at the time of covering e_i and the worst price would be c_j.
Therefore $\sum q_c \leq c_j H_k \leq c_j H_g$, which yields $\ln g$ ratio. ⊞

Algorithm 18 (LP-SC)**.** Solve LP given by min $\sum x_i c_i, \forall i \in [n] : \sum_{i \in S_j} x_j \geq 1$ over $x_i \geq 0$. $I := \{i : x_i^* \geq 1/f\}$

Theorem 18 (LP-SC is f-apx)**.** LP-SC is a f-approximation algorithm.

Důkaz. $\sum_{j \in I} c_j \leq f \cdot \sum c_j x_j^*$
Also e is covered, as there must exist a $S_i : x_i^* \geq 1/f$. ⊞

Algorithm 19 (PRIMAL DUAL-ALG)**.** $y_1, \ldots, y_n = 0, I := \emptyset, E := \emptyset$
choose an uncovered edge arbitrarily, while it exists, $\delta : min_{i:e \in S_i}(c_i - \sum_{e' \in S_i} y_{e'})$, $y_e := y_e + \delta$, $I = I \cup \{i\}$ $\forall i : e \in S_i, c_i = \sum_{e' \in S_i} y_{e'}, E = \bigcup_{i \in I} S_i$

Theorem 19 (PRIMAL-DUAL is f-apx)**.** PRIMAL-DUAL is a f-approximation algorithm.

Maximal independent set

Input: $G = (V, E)$
Output: $I \subseteq V : G[I] \simeq I_{|I|}$ and I is maximal with respect to inclusion

Algorithm 20 (PARA-MIS)**.** for all v in parallel: if $\deg(v) = 0 : I := I \cup \{v\}, V := V \setminus \{v\}$
for all v in parallel: mark v with probability $1/(2\deg(v))$
for all edges uv in parallel: if both u, v are marked, then unmark the vertex of smaller degree (same degree - one randomly)
$S :=$marked vertices, $I := I \cup S, V := V \setminus (S \cup N(S))), E := E \cap \binom{V}{2}$

Definition 5 (PARA-MIS: good/bad vertices and edges)**.** A vertex is good, if it has more than $\deg(v)/3$ of neighbours with at most $\deg(v)$ and is bad otherwise. Denote the set of bad vertices by B.
An edge uv is good if at least one of u, v is good and is bad otherwise. Denote the set of bad edges by E_B.

Lemma 1 (L1)**.** $\exists \alpha > 0 : \forall v : v$ good $\Rightarrow P[v \in S \cup N(S)] \geq \alpha$

Důkaz. If v is good, then $P[\exists w \in N(v) : w$ is marked$] = 1 - \prod_{w \in N(v)}(1 - \frac{1}{2\deg(w)}) \geq 1 - \prod_{w \in N(v), \deg(w) \leq \deg(v)}(1 - \frac{1}{2\deg(w)}) \geq 1 - (1 - \frac{1}{2d_v})^{\frac{d_v}{3}} \geq 1 - e^{-1/6}$
$P[w \in S | w$ marked$] \geq 1 - \sum_{x \in N(w), \deg(x) \geq \deg(w)} \frac{1}{2\deg(x)} \geq 1 - \sum_{x \in N(w)} \frac{1}{2\deg(x)} \geq 1 - 1/2 = 1/2$
Therefore $\alpha = (1 - e^{-1/6}) \cdot \frac{1}{2}$. ⊞

Lemma 2 (L2)**.** $|E_B| \leq |E|/2$

Důkaz. Direct the graph: $e \in E$ is now oriented towards the higher-degree vertex.
A bad vertex does not have many incoming edges: t incoming, $> 2t$ outgoing. Hence $v \in V : \deg^+(v) \leq \deg^-(v)/2$.
Therefore $|E_B| \leq \sum_{v \in B} \deg^+(v) \leq \frac{1}{2} \sum_{v \in V} \deg^-(b) \leq \frac{1}{2}|E|$. ⊞

Theorem 20 (Phases of PARA-MIS)**.** The expected number of phases of PARA-MIS is $\mathcal{O}(\log n)$

Důkaz. Let E_i be E after i phases.
$\mathbb{E}[|E_{i+1}|] \leq \mathbb{E}[|E_i|] \cdot \beta : \beta := 1 - \alpha/2 < 1$, hence $\mathbb{E}[|E_i|] \leq m \cdot \beta^i \leq 1/2$ for $i = c \log n$. ⊞

Derandomization via pairwise independent variables

Proof of L1 with pairwise independence. $P[w \in S | w \text{ marked}] \geq 1/2$ still holds.
$P[\exists w \in N(v) : w \text{ marked}] = 1 - \prod_{x \in N(v)} (1 - p_x) \geq \sum_x p_x - \frac{1}{2} \sum_x \sum_y p_x p_y = \sum p_x (1 - \sum_{y \neq x} p_y/2) \geq 1/8$. ⊞

Matrix multiplication testing

Test $AB = C$ in $\Omega(n^2)$

Algorithm 21 (MMT)**.** Take $r \in \{0,1\}^n$ at random, Test $ABr = Cr$.

Theorem 21 (MMT is fast and reliable)**.** MMT can be implemented in $O(n^2)$ arithmetic operations and $P[MMT \text{ has a false positive}] \leq 1/2$.

Důkaz. First is simple: $A(Br), Cr$.
Second: $D = AB - C$, false positive: $D \neq 0_n$ and $Dr = o, X := Dr$.
$d_{ji} \neq 0 : x_j = \langle r, d_j \rangle = \sum d_{ji} r_j = \ldots + d_{ji} \cdot r_j + \ldots$. Fix $r_l : l \neq j : P[x = o] \leq P[x_j = 0] \leq 1/2$. ⊞

Polynomial identities testing

p, q polynomials, Question: $pq \stackrel{?}{=} r$, alternatively: $pq - r \equiv 0$?.

Algorithm 22 (POLYTEST)**.** p of (total) degree d, field K. Take $S \subset K$.

Take $r \in S^n$ randomly, test $p(r) = 0$

Lemma 3 (Polynomial and testing)**.** Let $p(x_1, \ldots, x_n)$ be a polynomial of total degree at most d over field K, $P \neq 0$. Let $S \subseteq K, r_1, \ldots, r_n \in S$ taken independently at random. Then $P[p(r_1, \ldots, r_n) = 0] \leq d/|S|$

Důkaz. By induction on n, for $n = 1$ simple.
$n \geq 2 : P(x_1, \ldots, x_n) = x_1^k \cdot A(x_2 \ldots, x_n) + B(x_1, \ldots, x_n)$
$P[A(r_2, \ldots, r_n) = 0] \leq \frac{d-k}{|S|}$, $P[P(r) = 0 | A(r_2, \ldots, r_n) \neq 0] \leq k/|S|$, hence $P[P(r) = 0] \leq d/|S|$. ⊞

Parallel algorithm for perfect matching in bipartite and general graphs

Definition 6 (Edmonds matrix)**.** Edmonds matrix of a bipartite graph with two n-sized partitions is a matx $B : b_{ij} = x_{ij}$ if there exists an edge u_i, v_j. 0 otherwise.

Theorem 22 (Edmonds matrix and its properties)**.** For G bipartite, B its Edmonds matrix:

1. $\det(B) \neq 0 \Leftrightarrow G$ has a perfect matching

2. $\text{rank}(B)$ =size of maximum matching

Důkaz. 1 simple
2: given a maximum matching, we can build a submatrix with nonzero determinant, hence $rank \geq size$ and knowing the rank it is the largest k scuh that there is a $k \times k$ submtx with nonzeor determinant, hence $rank \leq size$ from 1. ⊞

Fact 1 (Det computation)**.** The determinant cen be computed in time $O(\log^2(n))$ on $O(n^{3.5}k)$ machines, where n is the number of orws and k is the number of bits of entries of B.

Lemma 4 (The Isolating lemma)**.** Let $S_1, \dots, S_n \subset \{a_1, a_m\}$, let $w : \{a_1, a_m\} \to R \subseteq \mathbb{R}$ such that $|R| = r$ and w is uniformly random. Then $P[\exists i : w(S_i)$ is minimal and unique$] \geq 1 - m/r$.

Důkaz. Let $e \in \{a_1, \dots, a_m\}$, $A = \{S_i : e \notin S_i\}, B = \{S_i \setminus \{e\} : e \in S_i\}, \min_A = \min\{w(S) : S \in A\}, \min_B = \min\{w(S) : S \in B\}$. Minimal weight is either $\min_A$ or $w(e) + \min_B$
If $w(S_a) = w(S_b)$ is minimal, then there eixsts $e \in S_a \triangle S_b$ such that $S_a \in A, S_b \setminus \{e\} \in B$
$P[\min_A = w(e) + \min_B] \leq 1/r$ - $\min_A, \min_B$ are fixed, there exists at most one value $v \in R$ such that $\min_B - \min_A = v$
Therefore $P[\exists e : \min_A = w(e) + \min_B] \leq m/r$ via union bound. ⊞

Definition 7 (Tutte matrix)**.** A Tutte matrix of a graph on n vertices is a $n \times n$ matrix B, with entries $b_{ij} = x_{ij}, b_{ji} = -x_{ij}$ iff ij is an edge and 0 otherwise.

Notation: m is a monomial in Tutte matrix, $F(m)$ is a multilist of corresponding edges.

Lemma 5 (Monomial in Tutte mtx det)**.** If m has a nonzero coefficient in $\det(B)$, then $F(m)$ is a collection of even cycles covering all vertices.

Důkaz. Take a permutation corresponding to m. If it had an odd cycle, we may revert it and we would obtain the opposite coefficient, making it zero. ⊞

Algorithm 23 (PPM)**.**

1. $M = \emptyset, w$ a random function: $E \to [2m]$
2. C :=Tutte matrix after substitution $x_{uv} = 2^{w(uv)}$
3. compute $\det(C)$ and the largest $W \in \mathbb{Z} : 2^W | \det(C)$
4. $\forall uv \in E$ in parellel, compute $D := \det(C^{(u,v)})$ ($C^{(u,v)}$ is a matrix created by deleting rows and columns correspoding to u, v)
5. Put uv in M if $D \cdot 2^{2w(uv)}/2^w$ is an odd integer (iff the reciprocal is the highest power of two s.t. it divides the determinant)
6. Check if M is a perfect matching

Lemma 6 (Tutte matrix)**.** Let C be the substituted Tutte matrix. Then

1. if 2^W is the largest power of 2 dividing $\det(C)$, then G has a perfect matching M with $w(m) \leq W/2$.
2. if G has a unique min-weight matching M, $W = 2w(M)$, then
   (a) 2^W is the largest power of 2 dividing $\det(C)$
   (b) $\forall uv \in E : e \in M \Leftrightarrow E^{W-2w_{uw}}$ is the max-power of 2 dividing $\det(C^{(u,v)})$

Důkaz. 1) $\exists m : W(F(m)) \leq W$. Then $F(m)$ is the set of even cycles, partition into $M_1 \sqcup M_2 := F(m)$, and one of them has weight $\leq w/2$.
2) a) perfect matching has a monomial with weight 2^W, via 1: we have a perfect matching
By contradiction: let there be two $\pm 2^W \pm 2^W : W = 2 \cdot 2^{w(M)} \leq 2^{w(M_1)} + 2^{w(M_2)} \leq 2^W$
b) follows by using a for $G[V \setminus \{u, v\}]$. If it weren't, we would get a better matching or a non-unique one. ⊞

List of topics

1	Algorithm (Average salary protocol)	1
1	Definition (Optimisation problem)	1
2	Definition (NP-optimisation problem)	1
3	Definition (Optimal value, approximation ratio)	1
1	Theorem (TSP nonapproximability)	1
4	Definition (Metric space)	1
2	Algorithm (TSP-MST)	2
2	Theorem (TSP-MST is 2-apx)	2
3	Algorithm (CHRISTOFIDES)	2
3	Theorem (CHRISTOFIDES is 1.5-apx)	2
4	Algorithm (QS)	2
5	Algorithm (ACCESS)	2
4	Theorem (ACCESS works whp in $2en \ln n$ rounds)	2
6	Algorithm (RANDMINCUT)	3
5	Theorem (RANDMINCUT and global minimum)	3
7	Algorithm (LOCAL SEARCH for SCHEDULING)	3
6	Theorem (LOCAL SEARCH is 2-apx)	3
8	Algorithm (GREEDY SCHEDULING)	3
7	Theorem (GREEDY is also 2-1/m-apx)	3
	Remark (Competitive ratio)	3
8	Theorem (GREEDY competitive ratio is $2 - 1/m$)	4
9	Algorithm (LARGEST PROCESSING TIME)	4
9	Theorem (LPT is 4/3-apx)	4
10	Algorithm (FIRST/BEST/ANY FIT BIN PACKING)	4
10	Theorem (ANYFIT is 2-apx)	4
	Remark	4
11	Algorithm (GREEDY EDP)	4
11	Theorem (GREEDY EDP is a $2\sqrt{m} + 1$-apx)	4
12	Algorithm (GREEDY EDP capacities)	5
12	Theorem (GREEDY is $\mathcal{O}(m^{1/(c+1)})$-apx)	5
13	Algorithm (RAND-SAT)	5
13	Theorem (RAND-SAT is a 2-apx)	5
14	Algorithm (BIASED-SAT)	5
14	Theorem (BIASED-SAT is $\phi - 1$-apx)	5
15	Algorithm (LP-SAT)	5
15	Theorem (LP-Sat is a $1 - 1/e$-apx)	5
16	Algorithm (BEST-SAT)	5
16	Theorem (BEST-SAT is a 3/4-apx)	5
17	Algorithm (GREEDY-SC)	6
17	Theorem (GREEDY-SC is $\ln g$-apx)	6
18	Algorithm (LP-SC)	6
18	Theorem (LP-SC is f-apx)	6
19	Algorithm (PRIMAL DUAL-ALG)	6

19 Theorem (PRIMAL-DUAL is f-apx) . . . 6
20 Algorithm (PARA-MIS) . . . 6
5 Definition (PARA-MIS: good/bad vertices and edges) . . . 6
1 Lemma (L1) . . . 6
2 Lemma (L2) . . . 6
20 Theorem (Phases of PARA-MIS) . . . 7
21 Algorithm (MMT) . . . 7
21 Theorem (MMT is fast and reliable) . . . 7
22 Algorithm (POLYTEST) . . . 7
3 Lemma (Polynomial and testing) . . . 7
6 Definition (Edmonds matrix) . . . 7
22 Theorem (Edmonds matrix and its properties) . . . 7
1 Fact (Det computation) . . . 8
4 Lemma (The Isolating lemma) . . . 8
7 Definition (Tutte matrix) . . . 8
5 Lemma (Monomial in Tutte mtx det) . . . 8
23 Algorithm (PPM) . . . 8
6 Lemma (Tutte matrix) . . . 8