WYKŁAD

Organizacyjnie

Przechodzę na „Ty". Moje dane. Zrobię stronę.

Parę słów ogólnie o wykładzie:

- trochę taka popularyzacja wyników
- idea: poznać to, co jest ważne w IT (inf. teor.), nagroda Gödla jako wyznacznik
- raczej wykład dla pasjonatów, będę starał się trzymać luźną formę
- na wielu tematach nie znam się dobrze, jak ktoś chce coś dodać, to b. dobrze
- w programie były 2 rzeczy, które są wykładane na Złożoności obliczeniowej: NL=coNL i IP=PSPACE. Ominiemy je, a za to zrobimy inne rzeczy, jeszcze myślę co.
- cel też: nabyć trochę wiedzy ogólnej o wynikach, ludziach

Ćwiczenia: będziemy uzupełniać dowody z wykładu, robić ciekawe zadania w okolicy, ew. oswajać się z teorią na następnym wykładzie

Końcowe pytania!

Zaliczenie: trzeba będzie sobie wybrać kilka (może 3-4) ulubione tematy i je umieć dobrze. Sprecyzuję to niedługo.

Na 2 pierwszych wykładach opowiem ogólnie o nagrodach Gödla, przejdę przez wszystkie.

Nagroda Gödla

Nagroda Gödla przyznawana jest od 1993 roku za wybitne osiągnięcia w dziedzinie informatyki teoretycznej.
Przyznają wspólnie EATCS (European Association for Theoretical Computer Science) i ACM SIGACT (Association for Computing Machinery Special ~~Group~~ Interest Group on Algorithms and Computation Theory). Nagroda jest wręczana albo na konferencji STOC (ACM) albo na konf. ICALP (EATCS). Rezultat musi być opublikowany w recenzjowanym czasopiśmie w ciągu ostatnich 14 (dawniej 7) lat.

Już były przyznane 22 nagrody. Średnio od publikacji mie 8 lat. Napiszę listę nagród, potem będę omawiał.
Jeśli chodzi o dziedziny, to podział jest następujący: (mój, bardzo umowny)

- złożoność obliczeniowa (8)
- algorytmy (6)
- inne, często zadziwiające połączenia dziedzin lub bardzo wpływowe nowe ~~idea~~, czasem proste, idee (8)

1993 rachunek dowodów interaktywnych
1994 dolne ograniczenia rozmiarów obwodów boolowskich
1995 twierdzenie IS: $NL = coNL$
1996 aproksymacja permanentu
1997 formalna def. wiedzy
1998 tw. Tody: $PH \subseteq P^{PP}$
1999 kwantowy alg. Shora
2000 logika te[illegible] ?

2001 tw. PCP
2002 równoważność dBPPA jest rozstrz.
2003 alg. AdaBoost
2004 aplikacje topologii w syst. rozpr.
2005 algorytmy strumieniowe
2006 test pierwszości AKS
2007 dowody naturalne
2008 analiza smoothed

2009 produkt zygzakowy i $L = SL$

2010 PTAS dla ETSP

2011 optymalne rezultaty nieaproksymowalności

2012 podstawy algorytmicznej teorii gier

2013 Diffie-Hellman dla wielu osób i Boneh-Franklin schemat

2014 optymalna agregacja

Teraz będę po kolei opowiadał parę słów o rezultatach

<u>1993</u> ~~[illegible]~~ Za rozwój dowodów interaktywnych

László Babai, Shlomo Moran, Shafi Goldwasser, Silvio Micali, Charles Rackoff

Wymyślili dowody interaktywne.

Idea: należenie do języka rozstrzyga się nie (jak zwykle) przez obliczenie maszyny Turinga, ale przez ~~[illegible]~~ ~~[illegible]~~ grę między dwoma graczami: Proverem i Verifierem.

Prover chce pokazać, że pewne słowo należy do języka, a Verifier go sprawdza.

Formalnie:

Język L należy do IP jeśli istnieje probabilistyczna MT V (Verifier) z czasem ograniczonym wielomianowo taka, że dla każdego w:

- $w \in L \Rightarrow \exists$ Prover P (MT): $\Pr[(P,V)(w) = 1] > \frac{2}{3}$
- $w \notin L \Rightarrow \forall$ Provera P (MT): $\Pr[(P,V)(w) = 1] < \frac{1}{3}$

istotne tylko, by były odgraniczone od $\frac{1}{2}$ i od 0 i 1.

Przykład:

Język par grafów ~~???~~ (G_1, G_2) t.że są nieizomorficzne

Prover		Verifier
		Zgadnij $i \in \{1,2\}$ oraz permutację $\pi \in S_n$, gdzie $n = \lvert G_1 \rvert = \lvert G_2 \rvert$. Niech $H = \pi(G_i)$
Znajdź j t.że $H \cong G_j$	$\xleftarrow{H}$	
	$\xrightarrow{j}$	~~Akceptuj~~ Odrzuć, gdy $i \neq j$.

powtórz kilka razy, po których prawd. $\frac{1}{2^k}$

Było jasne, że $NP \subseteq IP \subseteq PSPACE$.
Dużym wynikiem było pokazanie, że $IP = PSPACE$, czyli każdy język w PSPACE da się w ten sposób sprawdzić.

1994 Za prawie optymalne dolne ograniczenia dla obwodów boolowskich o małej głębokości.

Johan Håstad

Dawniej (przed wynikiem Razborova i Rudicha w 94, nagroda G. w 07, po więcej o tym) myślano powszechnie, że dobrym pomysłem na rozstrzygnięcie kwestii czy P=NP? jest badanie obwodów boolowskich. Gdyby dało się pokazać, że nie ma obwodu boolowskiego wielkości wielomianowej dla SAT-a, to by pokazywało, że $P \neq NP$. Takich ograniczeń dolnych nie dawało się pokazać dla dowolnych obwodów, ale ~~???~~ pracowano nad tym.

Håstad w swojej pracy pokazał, że dla obliczenia funkcji parzystości (~~...~~ parzystość sumy bitów, taki mega xor) jeśli używamy obwodów o głębokości k i bramek AND, OR i NOT, to wielkość musi być $\exp\left(\Omega\left(\sqrt[k-1]{n}\right)\right)$.
Ważniejszy był nawet lemat, którego użył, tzw. Håstad's switching lemma. Więcej ~~pracy~~ na wykładzie o dowodach naturalnych, czyli, że ta metoda ma swoje ograniczenia.

<u>1995</u> Za pokazanie, że NL = coNL

Neil Immerman, Róbert Szelepcsényi

Nim opowiem o wyniku to powiem o historii z nim związanej; zupełnie niezwykłej. Przez trzy dekady problem, czy NL = coNL był otwarty. NL to problemy rozwiązywalne przez niedet. maszyny Turinga z pamięcią logarytmiczną. ~~Pypo~~ coNL to ich dopełnienia. Typowy problem (NL-zupełny) to osiągalność w grafie (z s do t).
Alg.: zaczynamy w s, chodzimy n czasu (pamiętamy to w $\log n$ bitach), jak dojdziemy do t, to akceptujemy, jak nam się skończy czas to odrzucamy. Jest ścieżka z s do t $\iff$ istnieje obliczenie akceptujące. Czyli wystarczy pokazać, że „nie istnieje ścieżka z s do t" $\in$ NL. Szelepcsényi, student st. licencjackich w Bratysławie, chyba jako pracę licencjacką, pokazał to. To ~~jest~~ można zrobić na 2 stronach. Mimo to mógł to popracować itd., aż dowiódł tego też Neil Immerman, znany naukowiec. Jakoś udało się porozmawiać, że Sz. też to zrobił. On użył fajnej techniki liczenia.
Jak będziemy chcieli to zrobimy, ale jest na kursie ZO.

1996 Za aproksymację permanentu, prace nad Łańcuchami Markova

Mark Jerrum, Alistair Sinclair

Rozważmy macierz $M = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & & & \vdots \\ \vdots & & & \\ a_{n1} & \dots & \dots & a_{nn} \end{bmatrix}$

wyznacznik
$$\det(M) = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \cdot \prod_{i=1}^{n} a_{i\sigma(i)}$$

permanent
$$\mathrm{per}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)}$$

klasa złożoności, więcej przy tw. Toda

Wyznacznik da się obliczyć w czasie wielomianowym łatwo. Okazuje się, że permanent (chyba) nie. Jest tw. Valianta, że permanent jest #P-zupełny, czyli w szczególności jeśli $P \neq NP$ to nie ma wiel. alg. dla obliczania permanentu.

J. i S. pokazali, że można przybliżać perm. z dowolną dokładnością, tzn. $\forall_{\varepsilon > 0, \delta \in (0,1)} \exists$ OUT – wynik alg. wielomianowego.

$$\mathbb{P}\left[(1-\varepsilon)\,\mathrm{per}(A) \le OUT \le (1+\varepsilon)\,\mathrm{per}(A)\right] \ge 1-\delta,$$

Zrobili to technikami używającymi Łańcuchy Markova, ale nie wiem jak. Nie wgłębiałem się. Podobno te techniki potem stały się popularne.

1997 Za formalizację pojęcia wiedzy w systemach rozproszonych.

Joseph Halpern, Yoram Moses

Rozważmy taką zagadkę:

~~[illegible]~~

~~[illegible]~~

Dzieci bawiły się farbami.

Przychodzi tata i mówi, że niektóre z nich są umazane na twarzy. Jest n dzieci, które są niezwykle inteligentne. Tata pyta się dzieci:

– czy wiecie, czy macie umazaną twarz?

– dzieci odpowiadają (oddzielnie),

a potem sytuacja (pytanie, odpowiedź) powtarza się.

Jak to będzie wyglądało jeśli k dzieci jest brudnych?

~~[illegible]~~ Dla $k=1$ brudne dziecko powie: TAK w 1 turze. Wszystkie inne w drugiej.

Dla $k=2$ brudne dzieci powiedzą: TAK w 2 turze, reszta w trzeciej.

Ogólnie: w k-tej turze dzieci brudne się zgłoszą, czyste turę później.

Pozornie (dla $k>2$) tata nie wniósł informacji, wszyscy wiedzieli, że ktoś jest brudny. Jednak nieprawda, że wszyscy wiedzą, że wszyscy wiedzą, że ktoś jest brudny itd.

Halpern i Moses sformalizowali m.in. to pojęcie. Przyglądając się to

1998 Za to Toda, czyli, że $PH \subseteq P^{PP}$.

Seinosuke Toda

Najpierw co to za pojęcia. Mówiąc już dziesiątego rzeczy i wyroczni. $NP^{NP} = NP$, $NP^{coNP} = \Sigma_2^P$, $coNP^{NP} = \Pi_2$, $coNP^{coNP} = coNP$.

$NP^{\Pi_k^P} = \Sigma_{k+1}^P$, $coNP^{\Sigma_k^P} = \Pi_{k+1}^P$. Łatwo zobaczyć, że $\Pi_k^P \subseteq \Sigma_{k+1}^P$ i $\Sigma_k^P \subseteq \Pi_{k+1}^P$

To odpowiada alternacji: czy dla każdego x istnieje y t.że dla każdego z itd.

$$PH = \bigcup_{i=1}^{\infty} (\Sigma_k^P \cup \Pi_k^P) = \bigcup_{i=1}^{\infty} \Sigma_k^P = \bigcup_{i=1}^{\infty} \Pi_k^P.$$

polynomial hierarchy

P^{PP} to P z wyrocznią PP. Co to jest PP.

Chodzi o liczenie. Liczenie jest trudniejsze niż rozwiązanie, czy jest >0 rzeczy. #SAT to problem policzenia dla φ ile jest wartościowań spełniających. (czyli funkcja f: input $\to \mathbb{N}$).

#CYCLE – ile jest cykli prostych, #CYCLE $\notin$ FP o ile $P \neq NP$.

~~...~~ funkcje obliczalne przez det. ~~MT~~ w PTIME

~~#P~~ Funkcja ~~#P~~ $f: \{0,1\}^* \to \mathbb{N}$

należy do klasy #P jeśli istnieje wielomian $p: \mathbb{N} \to \mathbb{N}$ i PTIME MT M taka, że dla każdego $x \in \{0,1\}^*$ zachodzi

$$f(x) = \left| \{ y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1 \} \right|$$

liczba y pasujących do x

PP to decyzyjna wersja $\underbrace{\#P}_{\text{pytanie o najw. bit}}$, czyli $L \in PP$ jeśli istnieje PTIME MT M, wiel. $p: \mathbb{N} \to \mathbb{N}$ t.że $\forall_{x \in \{0,1\}^*}$

$$x \in L \iff \left| \{ y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1 \} \right| \geq \frac{1}{2} \cdot 2^{p(|x|)}$$

To [illegible]

1999

Za algorytm Shora faktoryzacji liczb na komputerze kwantowym

Peter Shor

Nie jest znany algorytm faktoryzacji (rozkładu na czynniki pierwsze) w czasie wielomianowym.
Nie są też znane żadne algorytmy losowe (ani heurystyczne, jak dobrze wiem), które działają sensownie.
To jest inaczej niż w przypadku pytania: czy liczba jest pierwsza? Jest znany alg. wiel. AKS (2004), duży wynik, jeszcze opowiem o nim, ale wcześniej znany był ~~alg~~ test Millera-Rabina, który używał losowości, ale w praktyce był bardzo dobry.
To, że nie ma efektywnych alg. dla faktoryzacji ma duże znaczenie w kryptografii: mnożenie nie może być odwrócone. Np. RSA na tym stoi. Druga funkcja, która jest uważana za trudną to logarytm dyskretny: znaleźć c t.że $a^c = b \mod n$.

W 1997 roku Peter Shor pokazał, że da się zrobić faktoryzację na ~~dla~~ komputerze kwantowym (tak samo logarytm dyskretny). Ten rezultat spowodował wielki rozwój badań nad komputerami kwantowymi. Niestety, naprawdę, chyba niewiele z tego wynikło w praktyce. Powiemy sobie jeszcze później dokładnie o alg. Shora.

2000 Za pracę nad logikami temporalnymi dla automatów skończonych

Moshe Vardi, Pierre Wolper

Głównym postępem było wprowadzenie technik automatowych do analizy logik temporalnych. Idea logik temporalnych jest taka, że służy do weryfikacji pewnych własności nieskończonych biegów programów.

Podstawowa taka logika to LTL. Zakładamy, że mamy nieskończone słowo (bez liter) etykietowane predykatami, czyli element $(2^P)^\omega$.

Formuły to: $p, \neg\varphi, \varphi\vee\psi, \varphi\wedge\psi, X\varphi, \varphi U \varphi$, też $F\varphi, G\varphi$.

Vardi i Wolper zaproponowali rozszerzenie, dodając do ETL ~~predykaty typu~~ formuły typu: $A(\varphi_1, \varphi_2, \varphi_3)$, gdzie A jest automatem nad $\Sigma = \{a_1, a_2, a_3\}$. $A(\varphi_1, \varphi_2, \varphi_3)$ jest prawdziwe, gdy akceptuje pewne słowo, w którym miejsca, gdzie zachodzi φ_i zamieniono na a_i (A - aut. Büchiego).

V. i W. argumentowali, że takie rozszerzenie jest przydatne, np. LTL nie wyrazi $(p,q)^\omega$. Pokazali też PSPACE alg. operujący się na automatach i odp., czy $\varphi \in ETL$ jest spełnialna.

Idea jest taka: dla każdej podformuły zgadujesz w każdym miejscu, czy jest ona prawdziwa, a potem sprawdzasz automatem Büchiego (wykładniczym), czy jest to ok.

Niepustość aut. B. jest w ~~[illegible]~~ NL, więc wychodzi NPSPACE = PSPACE

Ten automat pracuje nad alfabetem $2^{sub(\varphi)}$, tzn. dla każdej

2001 Twierdzenie PCP

Sanjeev Arora, Uriel Feige, Shafi Goldwasser, Carsten Lund, Laszlo Lovász, Ranjeev Motwani, Shmuel Safra, Madhu Sudan, Mario Szegedy

Twierdzenie PCP (Probabilistically Checkable Proofs) mówi, że $PCP(O(\log n), O(1)) = NP$.
$PCP(r(n), q(n))$ to klasa problemów, dla których świadkiem, że słowo należy do języka jest dowód, który może być sprawdzony przez weryfikatora przy użyciu $r(n)$ bitów losowych i $q(n)$ zapytań.
Gdy weryfikator mówi TAK, to $w \in L$, gdy mówi NIE, to $w \notin L$ z prawd. $\geq \frac{1}{2}$.

Ta nowa charakteryzacja łączy nieco idee dowodów interaktywnych i losowości. Poza tym otwiera zupełnie nowe horyzonty, pozwala rozumieć lepiej czym jest NP.
Przy pomocy tego tw. daje się też dowodzić, że pewnych problemów nie da się aproksymować z dobrą stałą. O tym jeszcze będziemy mówić.

2002 Za pokazanie, że równoważność deterministycznych automatów ze stosem jest rozstrzygalna.

Gérard Sénizergues

Znanym faktem jest, że równość dwóch języków bezkontekstowych (danych jako gramatyki lub ~~...~~ automaty ze stosem) jest nierozstrzygalna. Sénizergues pokazał w 2001, że dla tak zwanych deterministycznych automatów ze stosem ten sam problem jest rozstrzygalny. To pytanie było otwarte od 1966 roku. Aut. ze stosem jest deterministyczny jeśli:

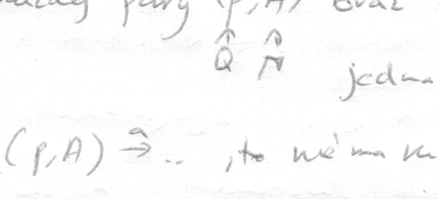
— dla każdej pary (p, A) $\hat{Q}\ \hat{A}$ oraz $a \in \Sigma \cup \{\varepsilon\}$ jest tylko jedna tranzycja $(p,A) \xrightarrow{a} (q,\alpha)$

— jeśli $(p,A) \xrightarrow{a} \ldots$, to nie ma ruchów $(p,A) \xrightarrow{\varepsilon} \ldots$ i odwrotnie

Moja uwaga osobista: „prawdziwą" przyczyną, dla której to się udało jest taka, że relacja bisymulacji jest rozstrzygalna dla automatów ze stosem. Tak się składa, że dla deterministycznych systemów (czyli też dPDA) bisymulacja i równoważność języków są tym samym.

to pokazano niedługo po Sénizergues'ie

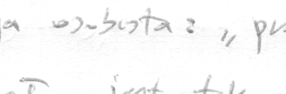
Idea bisymulacji: liczy się rozgałęzienie.



$p \not\sim q$

(gracz Spoiler i Duplikator)

ale



D-d S. był b. długi i skomplikowany. Niedawno Jancar opublikował nowy. Bisymulacją zajmowałem się w czasie

2003 Za algorytm AdaBoost

Yoav Freund, Robert Shapire

Algorytm AdaBoost pozwala na stworzenie dobrego klasyfikatora z wielu słabych klasyfikatorów.

Klasyfikator to ~~...~~ funkcja dzieląca zbiór obiektów na grupy (klasyfikująca). ~~Np. zadajemy pytanie: czy iść na studia na uniwersytet X. Mamy dużo kryteriów: czy ma dobrą opinię, czy jest blisko~~

Np. zadajemy pytanie: w co zainwestować na giełdzie? Mamy wiele przesłanek: czy spółka ma dobre oceny specjalistów, czy jej akcje ostatnio szły w górę, co mówią znajomi o tej firmie, czy sam bym kupił produkt tej firmy itd.

Można to zamodelować jako zbiór funkcji

$$f_i : X \to \{-1,+1\}$$ (mówimy TAK = +1 lub NIE = -1)

Idea jest taka, żeby zrobić dobry klasyfikator postaci

$$f = \sum_{i=1}^{n} \alpha_i f_i \quad \text{, gdzie } \alpha_i \in \mathbb{R} \quad , f: X \to [-1,1]$$

i jest dobrym klasyfikatorem.

AdaBoost — adaptive boosting. Boosting odpowiada temu, że klasyfikator jest sumą poprzednich.

Adaptive odpowiada temu jak AdaBoost działa.

On ~~bierze~~ wybiera pewien słaby klasyfikator i dodaje mu wagę.

Potem wybiera drugi (optymalnie) i dobiera (optymalnie) wagę itd.

To są właśnie te dwie rzeczy, które optymalizuje.

Jest prosty, ale bardzo się przydaje w praktyce, co łatwo sobie wyobrazić.

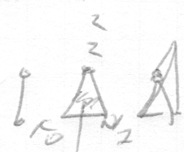
2004 Za aplikację topologii do teorii systemów rozproszonych

Maurice Herlihy, Michael Saks, Nir Shavit, Fotios Zaharoglou

Przeformułowanie problemu pozwoliło, przy użyciu technik z topologii, rozwiązać problem. Ogólnie rzecz biorąc te techniki przydają się do pokazania, że pewne rzeczy nie dadzą się zrobić. Pokażę to na przykładzie.

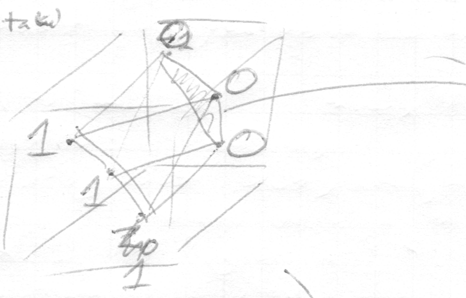
Rozważmy 3 procesy, Zielony, Czerwony, Niebieski, każdy ma 1 bit: 0 lub 1. To jest problem konsensusu, chcą ustalić taki sam bit na koniec. Jeśli na początku mieli taki sam, to powinni mieć to na końcu.

Reprezentujemy sytuację za pomocą zbioru sympleksów:

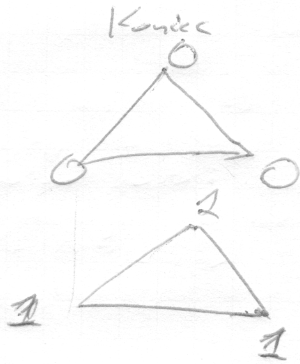


To oznacza, że Z ma daną 2, C ma 0, N ma 1.

Początek jest taki

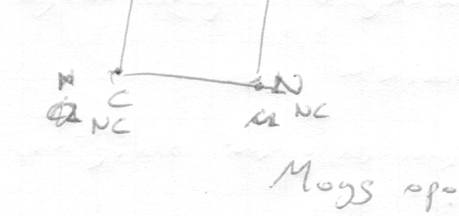


Koniec

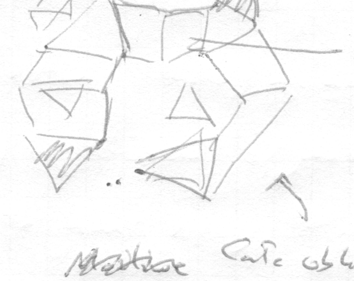


← Wiele ścieżek

To się zmienia w czasie, Procesy mogą się psuć. Np.



po 1 turze



jest świeczka

~~Możliwe~~ Całe obliczenie

Mogę opowiedzieć o tym więcej.

2005 Za wkład w algorytmy strumieniowe

Noga Alon, Yossi Matias, Mario Szegedy

Wyobraźmy sobie strumień danych $a_1, a_2, a_3 \ldots a_m$, gdzie $\forall_i \; a_i \in \{1, 2, \ldots, n\}$. Niech $m_i = |\{j : a_j = i\}|$, czyli ile razy wystąpiła liczba i. Definiujemy $F_k = \sum_{i=1}^{n} m_i^k$, k-ty moment. F_0 to liczba różnych elementów, F_1 to n, F_∞^* (zdef. oddzielnie) to maksymalna wartość m_i (bo to jest nowe ∞). Łatwo policzyć takie normy po prostu zliczając jakie są m_i, do tego potrzeba jednak liniowej pamięci. Wtedy dużo czasu zajmują operacje dostępu albo modyfikacji. Dlatego postawionym celem jest użycie jak najmniejszej ilości pamięci przy jednokrotnym przejrzeniu strumienia. Jednocześnie jest zgoda na przybliżenie (używamy losowości). Autorzy pokazali, że F_0, F_1, F_2 dadzą się przybliżyć w logarytmicznej pamięci, ale F_k dla $k \geq 6$ już nie. Pokazali, że

z dokł. do pewnego stałego czynnika, z zad. prawd.

F_k zawsze można aproksymować używając $O(n^{1-\frac{1}{k}} \log n)$ bitów, a dla $k \geq 6$ potrzeba $\Omega(n^{1-\frac{5}{k}})$ bitów.

Najciekawszy rezultat to ten, że F_2 da się zrobić w $O(\log n)$ bitów. Idea: $X_1, \ldots, X_n$ – rademachery, czyli $\mathbb{P}(X_i = 1) = \mathbb{P}(X_i = -1) = \frac{1}{2}$. Niech $X = \sum_{i=1}^{n} X_i m_i$. $\mathbb{E}(X^2) = \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i m_i\right)^2\right] = \ldots = \sum_{i=1}^{n} m_i^2$. X_i da się wyliczać dynamicznie (z pewnego skompresowanego systemu), więc to można zrobić w pamięci $O(\log n)$.

2006 Za algorytm AKS testowania pierwszości

Manindra Agrawal, Neeraj Kayal, Nitin Saxena

Autorzy opracowali pierwszy deterministyczny (nie używający losowości) algorytm testujący, czy liczba na wejściu jest pierwsza w czasie wielomianowym (od wielkości wejścia, czyli liczby cyfr). Wcześniej istniały dobrze działające w praktyce algorytmy testujące pierwszość (np. test Millera-Rabina), przy czym one używały losowości. A więc AKS nie przyniósł przełomu w praktyce, jednak jest to wielki postęp w teorii.

Ciekawe, że praca ma tylko 9 stron (aktualna wersja, pierwotna chyba podobnie), z czego 2 to wstęp, a ok. 2 to bibliografia i podziękowania. Dowód prawie nie używa głębokiej matematyki, a algorytm jest bardzo prosty. Poprawiona wersja działa w około $O(\log^6 n)$, czyli nie tak źle, ale test MR działa w $O(n^3)$.

Zajmiemy się tym dokładnie za kilka wykładów.

2007 Za dowody naturalne

Alexander Razborov, Steven Rudich

Praca pokazuje, że pewnego typu technikami $\underbrace{\text{nie da się}}_{}$ najprawdopodobniej ~~[illegible]~~ udowodnić, że $P \neq NP$. Przez wiele lat ludzie wierzyli, że linia badań dolnych ograniczeń dla obwodów boolowskich może doprowadzić do rozwiązania problemu, czy $P \stackrel{?}{=} NP$. Te techniki polegały na tym, że obwód z pokazywanej klasy (ograniczonej na razie) klasy, który ma rozpoznawać pewien język musi być duży (w jakimś sensie). Gdyby udało się pokazać, że np. obwód dla SATa w ogóle musi być duży, to by dowodziło, że $P \neq NP$. W tym stylu był też dowód Håstada, za który dostał nagrodę Gödla w 1994 roku.

Autorzy pokazali, że tego typu technikami nie odpowiemy na pytanie o $P \stackrel{?}{=} NP$ pod warunkiem, że istnieją generatory pseudolosowe. ~~Generatory pseudolosowe~~ Powszechnie wierzy się, że generatory pseudolosowe istnieją. Taki generator to ziarno i det. alg. generujący następny stan i bit. Ciąg bitów ma być nieodróżnialny od losowego za pomocą MT z PTIME (chyba), lub jakoś tak.

Zajmiemy się tym dokładniej później.

2008

Za analizę smoothed

Daniel Spielman, Shanghua Teng

Typową analizą czasu działania algorytmu jest czas pesymistyczny.
Czasem jednak czas pesymistyczny jest zły, ale w praktyce algorytm działa dobrze (np. quicksort). Po to wprowadzono czas średni, jednak czas średni też nie jest idealny. Trzeba mieć rozkład wejścia, przy tym w pewnych warunkach może dawać złe czasy. S. i T. wprowadzili „coś pośrodku", analizę smoothed i pokazali, że algorytm ~~simplex dla~~ sympleks rozwiązujący zadania programowania liniowego ma wielomianową złożoność smooth ($O(n^3 \log(n/\sigma))$).

Idea jest taka: bierzemy input, zaburzamy go trochę i patrzymy jaka jest średnia złożoność problemu w tym otoczeniu. Złożoność smoothed to maksimum po wejściach z tych średnich. Założenie zaburzenia jest naturalne, w szczególności dla progr. lin. (pewna nieścisłość, maksymalizujemy liniową funkcję na niej). Zaburzamy o czynnik gaussowski.

smoothed value

$$sv(f) = \max_{\hat{x}} \mathbb{E}\left[f(\hat{x} + \|\hat{x}\| \cdot g) \right], \text{ gdzie } g \sim N(0, \sigma^2)$$

Bierzemy teraz za f funkcję czasu działania i mamy złożoność smoothed.
Potem ludzie robili sporo badań na ten temat.
Będziemy o tym jeszcze mówić.

2009

Za produkt zygzakowy i $SL = L$

Omer Reingold, Salil Vadhan, Avi Wigderson

Ekspander to graf, który ma stały stopień, ale w którym każdy mały zbiór wierzchołków (minimum k) ~~sąsiadów ma~~ ma wielu sąsiadów ($\geq \alpha \cdot k$, α-stałe). ~~Mają~~ Ekspandery mają wiele zastosowań w informatyce teoretycznej. ~~Ekspandery~~ Można pokazać, że losowy graf jest ekspanderem (pewne z dużym prawd.), ale jeśli chcemy stosować ekspandery do derandomizacji, to to nie zadziała. Chcemy umieć konstruować ekspandery deterministycznie. Przy użyciu produktu zygzakowego da się konstruować takie, produkując wybrane grafy można skonstruować wysoki ekspander o bardzo dobrych własnościach.

Wiadomo, że osiągalność na grafach skierowanych jest w NL, a do tego NL-zupełna. Na grafach nieskierowanych może to być łatwiejsze (szczególny przypadek). Tak długo to badano, że stworzono specjalną klasę: SL problemów redukowalnych do nieskierowanej osiągalności. Omer Reingold, używając ~~zygzaka~~ zygzaka pokazał w 2005 roku algorytm det. w log. pamięci ~~dla~~ nieskierowanej osiągalności, czyli ~~...~~ w szczególności, że $SL = L$.

Idea: po n^2 wywołaniach po $d^3 n^3$ krokach mamy już dużą szansę.

Jak to zderandomizować? Będziemy o tym mówić dokładnie.

2010 Za niezależne odkrycie PTAS dla ETSP

Sanjeev Arora, Joseph Mitchell

drugi raz n. G., autor książki o TZ

Najpierw o ETSP, PTAS to schemat aproksymacji.
TSP = travelling salesman problem.

TSP jest NP-zupełny. Co więcej TSP nie da się aproksymować (np. ze stałym czynnikiem, ale też ogólniej) w PTIME o ile P $\neq$ NP. To można pokazać przez redukcję do cyklu Hamiltona, robię G' jako: krawędzie w G z wagą 1, oprócz tych z wagą $3n$. Jeśli 2-aproksymacja nie zaw. kraw. spoza G, to jest c.H., jeśli zawiera to nie ma c.H.

Dlatego wprowadzono metryczny TSP, gdzie jest spełniona nierów. trójkąta. Da się go 2 aproksymować: robię drzewo rozpinające, przechodzę po nim i upraszczam.
Czy da się lepiej? Tak, da się $\frac{3}{2}$ $\rightarrow$ algorytm Christofidesa
(Tyczy drzewo rozpinające z doskonałym skojarzeniem).
Czy da się dowolnie dobrze?
Tzn., czy istnieje PTAS (polynomial time approximation scheme), dla każdego $\varepsilon > 0$ istnieje alg. wiel. aproksymujący z dokł. $(1+\varepsilon)$, może coraz wolniejszy dla mniejszych ε. Nie, okazuje się, że nie ma o ile NP $\neq$ P (rok 92). Hipoteza jest, że $\frac{4}{3}$ jest granicą.
Dlatego rozważono ETSP (euclidean TSP), na płaszczyźnie.
Okazuje się, że taki istnieje.

Zróbmy to później dokładniej.

2011

Za pokazanie optymalnych rezultatów nieaproksymacji

Johan Håstad

drugi raz n.G.

Praca pokazuje szereg rezultatów typu „problem X nie może być aproksymowany wielomianowym algorytmem lepiej niż co do czynnika α".
W szczególności jest to stała $\frac{8}{7}$ dla E3-SAT-a, która jest optymalna (spełnienie maks. liczby klauzul) oraz stała 2 dla E3-LIN-2, równań liniowych nad $\mathbb{Z}_2$ (czy F_2), która też jest optymalna.

Techniki są bardzo skomplikowane, ~~[illegible]~~ są podobne do technik stosowanych przy tw. PCP oraz dowodach interaktywnych. Ta praca pociągnęła za sobą też następne, jak to z UGC i rezultatem p. Oleszkiewicza, za który Subhash Khot dostał ostatnio nagrodę Nevanlinny.

Są tu też użyte (jak rozumiem po raz pierwszy w tej dziedzinie) techniki analizy fourierowskiej.

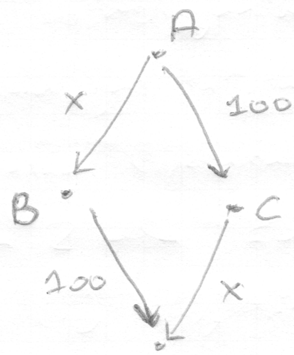
Możemy o tym powiedzieć więcej później.

2012

Za podstawy algorytmicznej teorii gier

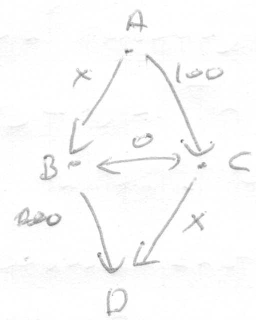
Elias Koutsoupias, Christos Papadimitriou, Noam Nisan, Amir Ronen, Tim Roughgarden, Éva Tardos

Rozważmy następujący przykład:



w sumie jest 100 kierowców

Średni koszt to 150. Teraz budujemy szybką drogę



— paradoks Braessa

Równowaga Nasha to 200 (albo 199). Pogorszyło się o $\frac{3}{4}$.

Jak bardzo tego typu rzeczy mogą się pogorszyć, gdy wagi na krawędziach są liniowe? Okazuje się, że właśnie $\frac{3}{4}$ jest optymalne.

Konkretnie centralnie sterowane optimum to $\frac{3}{4}$ RN. (pewnej)

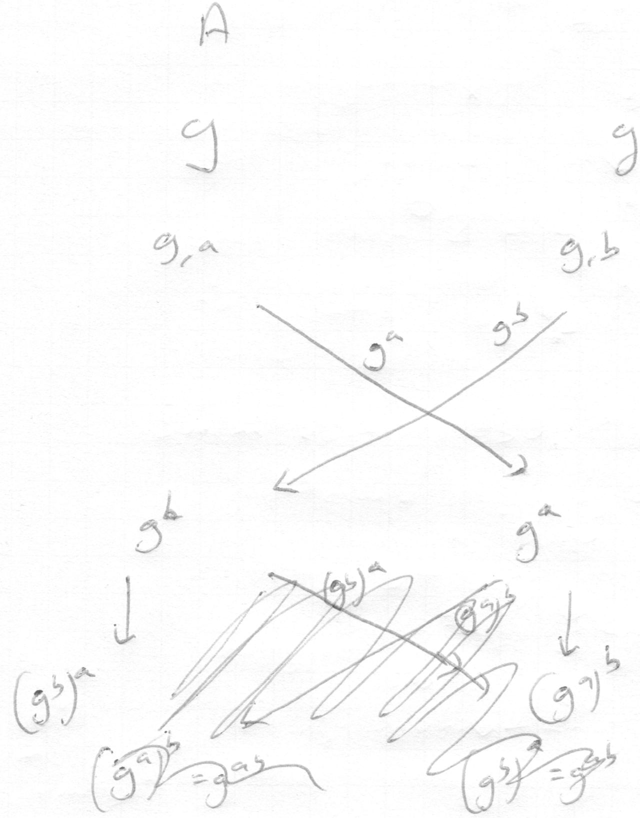
To się nazywa „price of anarchy"

Jeden artykuł to wprowadził, drugi pokazał dość pełny obraz (m.in. to $\frac{3}{4}$), trzeci pokazał aplikacje do tzw. „mechanism design", które jest zdaje się dziedziną pt. „jak zaprojektować system, by się ustalił w żądanych miejscach"

2013 – Za protokół Diffiego-Hellmana dla wielu agentów (i 1 tury) oraz za schemat Boneha-Franklina

Dan Boneh, Matthew Franklin, Antoine Joux

Protokół D-H wygląda normalnie tak: (A, B chcą ustalić wspólny klucz)



1. wybierają (wspólny) element $g \in \mathbb{Z}_p$
2. wybierają tajne $a, b \in \mathbb{N}$ (wówczas $a, b \le p-1$)
3. wysyłają sobie g^a, g^b
4. obliczają sobie $(g^b)^a, (g^a)^b$

Wspólny klucz to g^{ab}.

Trudność opiera się na fakcie, że dyskretny logarytm jest (chyba) trudny, tj. z $(g, g^a \bmod p)$ trudno odtworzyć a.

Teraz: da się to zrobić dla 3 (lub więcej) agentów. Robimy

$$(g,g,g) \to (g^a, g^b, g^c) \to (g^{ca}, g^{ab}, g^{bc}) \xrightarrow{\text{obliczamy}} (g^{bca}, g^{cab}, g^{abc}),$$

ale to wymaga 3 rund. Joux znalazł algorytm, w którym tylko jeden broadcast jest potrzebny.

(tu ciąg dalszy z poprzedniej strony)

Algorytm Joux jest oparty na krzywych eliptycznych, konkretnie parowaniu Weila (nie wiem co to jest). Ogólnie te krzywe eliptyczne (chyba) przydają się do zrobienia grupy o fajnych własnościach (lepszych niż $\mathbb{Z}_p$).

Boneh i Franklin opracowali schemat IBE też oparty na parowaniu Weila, który jako pierwszy (jak rozumiem) jest w pełni funkcjonalny. Taki schemat to generowanie kluczy prywatnych i publicznych w taki sposób, żeby kluczem publicznym był to znany ~~[illegible]~~ obiekt, np. email. To działa tak, jak chcę do X wysłać zaszyfrowaną wiadomość. Szyfruję to mailem X i wysyłam do X. Nie potrzebuję znać jego klucza publicznego. Gdy X chce odszyfrować, to zgłasza się do Instytucji, mówi, że chce klucz prywatny (i chyba też, że do tego maila) i oni mu go dają. ~~[illegible]~~ Problemem jest wiarygodność Instytucji, w każdym razie teoretycznie to już jest.

<u>2014</u> Za algorytm optymalnej agregacji dla middleware

Ronald Fagin, Amnon Lotem, Moni Naor

Rozważmy taką sytuację. Mamy (duży zbiór) obiektów, z których każdy ma ~~xxx~~ wiele (n) cech.
Np. jak duży jest, jak biały jest, jak kwadratowy jest itd.
Cechy są oznaczone liczbowo. Ocena każdego ~~ob~~ obiektu zależy od częściowych wartościowań i jest dana f-cją $f: \mathbb{N}^n \to \mathbb{N}$. Chcemy wybrać k obiektów o największej wartości f. Dla każdej cechy mamy listę posortowaną malejąco wartości obiektów. To można rozważać przy różnych paradygmatach (jak dostęp swobodny itd.).

Fagin dawniej (99) znalazł taki algorytm: (FA)
- leci po listach od góry, równocześnie
- każdy zobaczony na jakiejś liście obiekt odwiedź na innych listach i policz jego wartość
- jak znajdziesz k obiektów zobaczonych (w normalnym trybie) na wszystkich listach, to skończ i oblicz k najlepszych obiektów.

Ten alg. był dobry pod pewnym względem, ale potem znalazł lepszy. (threshold alg.), TA
- rób to samo co poprzednio
- trzymaj też „próg", czyli aktualną wartość funkcji od najniższych zobaczonych rzeczy
- jeśli jest k obiektów powyżej progu, to skończ

TA jest lepszy niż FA. Co ciekawe Fagin był w jakimś sensie zawstydzony prostotą algorytmu, na stronie ~~dodał~~ wręcz jego komentarz.