model Internetu, w którym węzły sieci są graczami, krawędzie – połączeniami, a użyteczność gracza (w tym przypadku koszt połączeń) jest mierzona długością i liczbą połączeń z innymi graczami. Autorzy przeanalizowali istnienie i własności równowag Nasha, przedstawili listę otwartych problemów, z których do dziś tylko kilka doczekało się rozwiązania. Otwartym problemem jest w szczególności pytanie, czy istnieją i jakie mają własności procesy dynamiczne, prowadzące do równowag (w tym i w późniejszych modelach tworzenia sieci).

Ważnym typem modeli teoriogrowych, stosowanych do wyjaśniania rzeczywistych zachowań, są modele uczenia, w których gracze zmieniają strategie w zależności od wypłat swoich i innych graczy, prowadzące do równowagi Nasha, np. „Replikator" czy „Gra fikcyjna" (*fictitious play*). Jednakże, m.in. w wyniku rozwoju ekonomii eksperymentalnej i po uwzględnieniu osiągnięć psychologii, stało się jasne, że ludzie zachowują się inaczej, niż przewidują modele uczenia.

Jednym z intrygujących problemów, który do tej pory nie ma powszechnie akceptowanego rozwiązania, jest zagadnienie ewolucji kooperacji w grupach ludzkich, w których interakcje są opisywane formalizmem TG. Najczęściej stosowanymi modelami takich sytuacji są *dylematy społeczne*, w szczególności „Dylemat Więźnia" i (w przypadku wielu graczy) „Dylemat Wspólnych Zasobów". Można przewidywać, że nowe modele będą w szerszym stopniu uwzględniały wzmiankowane wyżej aspekty behawioralne i neurobiologiczne takich interakcji.

Czytelnikowi pragnącemu rozszerzyć swoją wiedzę w obszarze problemów otwartych TG polecamy:

[1] A. Fabrikant et al., *On a network creation game*, PODC '03 (2003) 347-351;
[2] N. Nisan, T. Roughgarden, E. Tardos, V.V. Vazirani, *Algorithmic Game Theory*, (2007) Cambridge;
[3] Y. Shoham, K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, (2008) Cambridge Univ. Press;
[4] E. Elkind, K. Leyton-Brown, *Algorithmic Game Theory and Artificial Intelligence*, AI Magazine, 31, 4 (2010);
[5] M. Shubik, *The Present and Future of Game Theory*, Discussion Paper 1808 (2011);
[6] V. Fragnelli, G. Gambarelli, *Open problems in the theory of cooperative games*, IGTR, 15 2, 3 (2013);
[7] M.O. Jackson, Y. Zenou, Games on Networks, *Handbook of Game Theory with Economic Applications* (2015) Elsevier;
[8] E. Maskin, *How Can Cooperative Game Theory Be Made More Relevant to Economics?* Open Problems in Mathematics, J.F. Nash, Jr., M.Th. Rassias (eds.), Springer Int. Publ. (2016).

Mnożenie nie tylko macierzy

*Marcin MUCHA**

* Instytut Informatyki, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

Tematem tego artykułu jest mnożenie macierzy, ale zaczniemy od problemu nieco prostszego – mnożenia wielomianów. Niech A i B będą wielomianami stopnia n, $A(x) = \sum_{i=0}^{n} a_i x^i$, $B(x) = \sum_{i=0}^{n} b_i x^i$. Iloczynem A i B jest wielomian $C(x) = \sum_{i=0}^{2n} c_i x^i$, gdzie $c_i = \sum_{0 \leqslant k \leqslant i} a_k b_{i-k}$ (przyjmujemy $a_k = b_k = 0$ dla $k > n$).

Mnożenie wielomianów za pomocą tego wzoru wymaga rzędu n^2 operacji. Okazuje się jednak, że można to zrobić szybciej. Jako pierwszy zauważył to Anatolij Karatsuba już w 1960 roku. Algorytm Karatsuby opiera się na następującym spostrzeżeniu. Rozważmy iloczyn wielomianów szczególnie prostej postaci:

$$(a_n x^n + a_0)(b_n x^n + b_0) = a_n b_n x^{2n} + (a_n b_0 + b_n a_0) x^n + a_0 b_0.$$

Wydaje się, że do obliczenia współczynników wyniku potrzebne są cztery mnożenia, ale wystarczą trzy, gdyż

$$a_n b_0 + b_n a_0 = (a_0 + a_n)(b_0 + b_n) - a_0 b_0 - a_n b_n.$$

Co ciekawe, sztuczka zmniejszania liczby mnożeń była znana już Gaussowi, który używał jej do mnożenia liczb zespolonych za pomocą trzech mnożeń.

W ogólnym przypadku, aby pomnożyć dwa wielomiany A, B stopnia $2n-1$, zapisujemy A następująco:

$$A(x) = \sum_{0 \leqslant i < n} a_i x^i + x^n \Big(\sum_{0 \leqslant i < n} a_{i+n} x^i \Big),$$

i analogicznie B, a następnie korzystamy z wyprowadzonej właśnie tożsamości. Sprowadzamy w ten sposób jedno mnożenie wielomianów stopnia $2n-1$ do

Pracujemy teraz z wielomianami, których współczynniki same są wielomianami. Zachęcamy Czytelnika do sprawdzenia, że nie prowadzi to do żadnych nieprzewidzianych trudności.

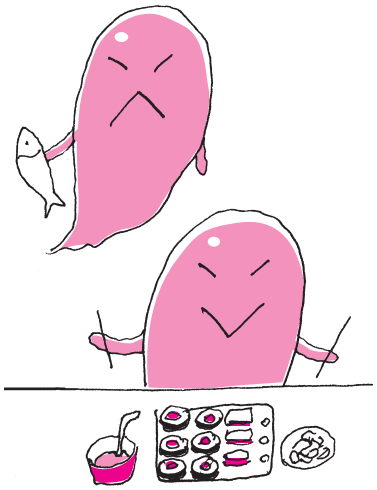
trzech mnożeń oraz dwóch odejmowań wielomianów stopnia $n-1$. Proces ten można kontynuować rekurencyjnie. Liczbę operacji arytmetycznych, wykonywanych przez uzyskany w ten sposób algorytm, opisuje równanie $T(2n-1) = 3T(n-1) + O(n)$, którego rozwiązaniem jest $T(n-1) = O(n^{\log_2 3})$, gdzie $\log_2 3 \approx 1{,}585$. Jeśli początkowe stopnie nie są postaci $2^k - 1$, należy je sztucznie zwiększyć, nie zmienia to rzędu liczby operacji.

Istnieją też inne efektywne algorytmy mnożenia wielomianów. Najszybszy z nich korzysta z szybkiej transformaty Fouriera i wymaga $O(n \log n)$ operacji arytmetycznych. Warto zwrócić uwagę, że algorytmów tych można używać także do mnożenia liczb, wstawiając za współczynniki wielomianów cyfry liczb, które chcemy pomnożyć. Poprawność tej procedury łatwo uzasadnić – wystarczy podstawić za x bazę systemu, w którym liczymy, np. $x = 10$. Zarówno algorytm Karatsuby, jak i inne szybkie algorytmy mnożenia liczb/wielomianów są używane w praktyce. Wiele bibliotek implementuje kilka różnych algorytmów i wybiera właściwy w zależności od rozmiaru mnożonych liczb.

Przejdźmy teraz do głównego tematu tego artykułu, czyli mnożenia macierzy. Niech A i B będą macierzami $n \times n$, $A = (a_{i,j})$, $B = (b_{i,j})$. Iloczynem A i B jest macierz $C = (c_{i,j})$, gdzie $c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$. Jak łatwo sprawdzić, mnożenie macierzy za pomocą tej definicji wymaga rzędu n^3 operacji. W 1969 roku Strassen zadziwił świat naukowy odkryciem algorytmu, który mnoży macierze za pomocą $O(n^{\log_2 7})$ operacji, gdzie $\log_2 7 \approx 2{,}81$. Algorytm ten opiera się na pomyśle podobnym do algorytmu Karatsuby. Strassen znalazł mianowicie metodę mnożenia dwóch macierzy 2×2 za pomocą 7 mnożeń zamiast 8, które wykonuje algorytm naiwny. Aby móc zastosować algorytm Strassena rekurencyjnie, wystarczy zauważyć, że jeśli każdą z macierzy A, B oraz $C = AB$ podzielimy na cztery identycznych rozmiarów kwadratowe bloki, to zachodzi:

$$\begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix} = \begin{pmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{pmatrix}.$$

Oznacza to, że można takie macierze blokowe mnożyć tak, jakby bloki były liczbami. W połączeniu ze sztuczką Strassena pozwala to zredukować mnożenie macierzy $(2n) \times (2n)$ do 7 mnożeń macierzy $n \times n$ i pewnej, zupełnie nieistotnej, liczby dodawań takich macierzy. Złożoność uzyskanego w efekcie algorytmu rekurencyjnego opisuje równanie $T(2n) = 7T(n) + O(n^2)$, którego rozwiązaniem jest wspomniana wcześniej złożoność $T(n) = O(n^{\log_2 7})$.



Po odkryciu Strassena sądzono, że znalezienie algorytmu o złożoności $O(n^2 \log n)$ lub podobnej jest tylko kwestią czasu. Stało się jednak inaczej. Przez kolejnych dwadzieścia lat rozwinięto niezwykle wyrafinowaną metodologię konstrukcji algorytmów mnożenia macierzy, wyrażoną w języku algebry tensorowej. W ramach tej teorii zaproponowano wiele algorytmów mnożenia macierzy, najszybszym z nich jest opublikowany w roku 1980 algorytm Coppersmitha i Winograda o złożoności $O(n^{2{,}38})$. Od tamtego czasu powstały nowe, dużo bardziej eleganckie teorie, m.in. interpretacja znanych algorytmów w języku działań grup. Parametry algorytmu Coppersmitha i Winograda zostały też zoptymalizowane, w efekcie uzyskano złożoność $O(n^{2{,}37})$. Pytanie o algorytm o złożoności niemal kwadratowej pozostaje nadal jednym z najbardziej fundamentalnych otwartych problemów algorytmiki.

Nie jest to jednak jedyny otwarty problem związany z mnożeniem macierzy. Mnożenie macierzy jest niezwykle często wykorzystywane w praktyce, choćby w implementacji warstw gęstych w tak ostatnio popularnych sieciach neuronowych. Pomimo to żadna z popularnych zoptymalizowanych implementacji nie korzysta z omawianych tu algorytmów. Powody ku temu są dwa. W przypadku najszybszych asymptotycznie algorytmów stała ukryta w notacji dużego O ma monstrualne rozmiary, przez co są one bezużyteczne w praktyce. Z kolei w przypadku wolniejszych asymptotycznie algorytmów, np. algorytmu Strassena, rekurencyjna struktura utrudnia efektywne zarządzanie pamięcią podręczną procesora (cache), które jest kluczowym elementem efektywnych implementacji. Czy istnieją naprawdę szybkie algorytmy mnożenia macierzy?