Najsłynniejsze algorytmy XX wieku

Adam Wójtowicz

Uniwersytet Warszawski

Proseminarium fizyki teoretycznej 6 marca 2006

Plan Wystąpienia.

Algorytmy poezją obliczeń.

Francis Sullivan

Algorytmy:

1. Stare jak cywilizacja:
   - Sumerzy,
   - Stonehenge.
2. Potrzeba matką wynalazków.
3. Zastosowanie:
   - komunikacja,
   - ochrona zdrowia,
   - przemysł,
   - ekonomia,
   - przewidywanie pogody,
   - nauki podstawowe.
4. Na czym się skupić?

10 Algorytmów XXw.

Wybór *Computing in Science & Engineering 2000.*

1. Metropolis Algoritm for Monte Carlo
2. Simplex Method for Linear Programming
3. Krylov Subspace Iteration Methods
4. The Decompositoinal Approach to Matrix Computations
5. The Fortran Optimizing Compiler
6. QR Algorithm for Computing Eigenvalues
7. Quicksort Algorithm for Sorting
8. Fast Fourier Transform
9. Integer Relation Detection
10. Fast Multipole Method

10 Algorytmów XXw.

Wybór *Computing in Science & Engineering 2000.*

1. Metropolis Algoritm for Monte Carlo
2. Simplex Method for Linear Programming
3. Krylov Subspace Iteration Methods
4. The Decompositoinal Approach to Matrix Computations
5. The Fortran Optimizing Compiler
6. QR Algorithm for Computing Eigenvalues
7. Quicksort Algorithm for Sorting
8. Fast Fourier Transform
9. Integer Relation Detection
10. Fast Multipole Method

Plan - przypomnienie.

Pasjans Ulama.

Stanisław Ulam, szpital w Los Angeles i solitaire

Jak policzyć wygrywające rozdania?

1. wylosujmy rozdanie.
2. czy jest ono wygrywające?
   - tak - $licznik := licznik + 1$,
   - nie - $licznik := licznik$.
3. wynik po M próbach to $\frac{licznik}{M}$.

Stan Ulam

Pasjans Ulama.

Stanisław Ulam, szpital w Los Angeles i solitaire

Jak policzyć wygrywające rozdania?

1. wylosujmy rozdanie.
2. czy jest ono wygrywające?
   - tak - $licznik := licznik + 1$,
   - nie - $licznik := licznik$.
3. wynik po M próbach to $\frac{licznik}{M}$.

Stan Ulam

Pasjans Ulama.

Stanisław Ulam, szpital w Los Angeles i solitaire

Jak policzyć wygrywające rozdania?

1. wylosujmy rozdanie.
2. czy jest ono wygrywające?
   - tak - $licznik := licznik + 1$,
   - nie - $licznik := licznik$.
3. wynik po M próbach to $\frac{licznik}{M}$.

Stan Ulam

Pasjans Ulama.

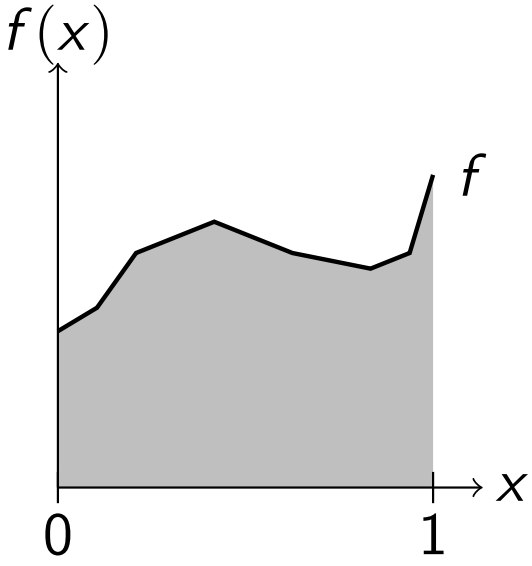
Stanisław Ulam, szpital w Los Angeles i solitaire

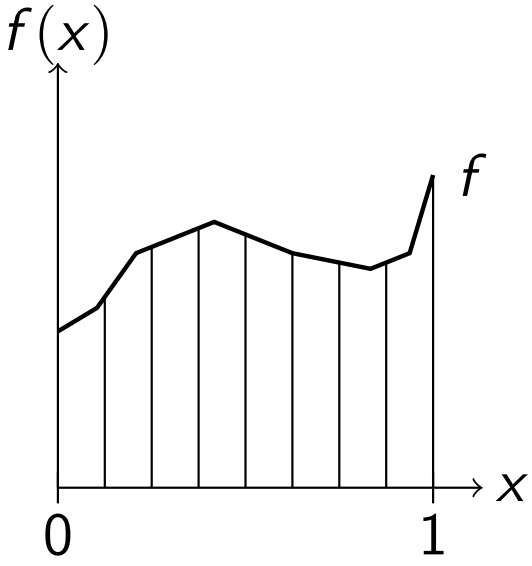
Jak policzyć wygrywające rozdania?

1. wylosujmy rozdanie.
2. czy jest ono wygrywające?
   - tak - $licznik := licznik + 1$,
   - nie - $licznik := licznik$.
3. wynik po M próbach to $\frac{licznik}{M}$.

Stan Ulam

Próbkowanie i całkowanie.





Obliczmy numerycznie całkę:

$$S = \int_0^1 f(x)dx.$$

Przybliżamy:

$$S \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n)$$

dzieląc odcinek $[0, 1]$ równomiernie na N punktów - metoda trapezów - zbieżność $1/N^2$ (dla całek d - wymiarowych $1/N^{2/d}$).
Liczby x_n wygenerowane losowo - metoda Monte Carlo - zbieżność $1/\sqrt{N}$.

Plan - przypomnienie.

Algorytm Metropolisa.

Obliczanie całki $S = \int_0^1 f(x)dx$

Często $f(x)$ nie gładka. Niekiedy część osobliwa daje się wydzielić jako gęstość pewnego *rozkładu prawdopodobieństwa* $p(x)$:

$$S = \int p(x)g(x)dx,$$

$$p(x) \geqslant 0 \quad oraz \int p(x)dx = 1.$$

Dla punktów $\{x_n\}$ generowanych z rozkładem $p(x)$ oszacowanie całki S dane jest średnią po trajektorii Monte Carlo:

$$S \approx \frac{1}{N} \sum_{n=1}^{N} g(x_n).$$

Algorytm Metropolisa.

Idea

Wprowdzenie procesu stochastycznego, generującego punkty x, których rozkład w granicy nieskończonej liczby kroków dąży do $p(x)$. Zwykle proces ten to Łańcuch Markowa o *prawdopodobieństwie przejść* $T(x \to x')$ spełniający warunek równowagi szczegółowej:

$$p(x)\,T(x \to x') = p(x')\,T(x' \to x).$$

Równanie Master

Jest to warunek dostateczny stałości w czasie prawdopodobieństwa o ewolucji opisanej równaniem:

$$p(x, t+1) = p(x, t) + \sum_{x'} [p(x', t)\,T(x' \to x) - p(x, t)\,T(x \to x')].$$

Zastosowanie w fizyce statystycznej.

Rozkład kanoniczny

Układ klasyczny w temperaturze T z oddziaływaniem $U(\mathbf{x})$ opisany jest prawdopodobieństwem:

$$p(\mathbf{x}) = \frac{1}{Q} exp(-U(\mathbf{x})/k_B T),$$

gdzie

$$Q = \int exp(-U(\mathbf{x})/k_B T) d\mathbf{x},$$

k_B stała Boltzmanna. Chcemy obliczać *wielowymiarowe* całki typu:

$$\langle A \rangle = \frac{1}{Q} \int A(\mathbf{x}) exp(-U(\mathbf{x})/k_B T) d\mathbf{x}.$$

Algorytm Metropolisa realizuje błądzenie przypadkowe z rozkładem prawdopodobieństwa $p(\mathbf{x})$.

Algorytm Metropolisa 1946.

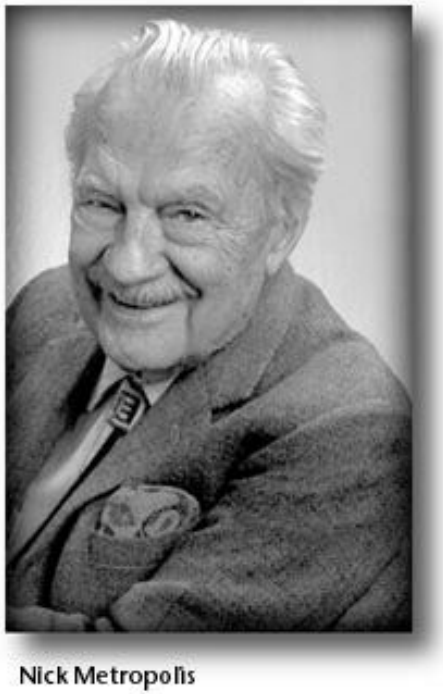
Warunek równowagi szczegółowej:

$$\frac{T(\mathbf{x} \rightarrow \mathbf{x}')}{T(\mathbf{x}' \rightarrow \mathbf{x})} = exp(-[U(\mathbf{x}') - U(\mathbf{x})]/k_B T),$$

ma rozwiązanie

$$T(\mathbf{x} \rightarrow \mathbf{x}') = min[1, exp(-[U(\mathbf{x}') - U(\mathbf{x})]/k_B T)].$$

Ten wybór nazywany jest algorytmem Metropolisa.



Nick Metropolis

Zastosowanie do modeli sieciowych.

Model Isinga.

$$H = -J \sum_{<i,j>}^{N} s_i s_j - B \sum_{i=1}^{N} s_i.$$

- $s_i = \pm 1$ - operatory spinowe,
- J oddziaływanie między niesparowanymi elektronami,
- B pole magnetyczne,
- $< i, j >$ sumowanie po najbliższych sąsiadach,
- N całkowita liczba węzłów.

$$\langle A \rangle = \frac{1}{Z} \sum_{s_i} A(s_i) exp(-H(s_i)/k_B T),$$

$$Z = \sum_{s_i} exp(-H(s_i)/k_B T),$$

suma po wszystkich (2^N) konfiguracjach spinów s_i.

Zastosowanie do modeli sieciowych.

Algorytm Metropolisa dla Modelu Isinga.

1. Wybierz stan początkowy (np. $s_i = 1$ dla $i = 1, 2, \ldots, N$).
2. Wybierz (losowo) węzeł i.
3. Oblicz zmianę energi ΔE gdy s_i zmienia wartość.
4. Wygeneruj liczbę losową r i taką, że $0 < r < 1$.
5. Jeżeli $r < exp(-\Delta E / k_B T)$ to zmianę akceptuj.
6. Idź do 2.

Częstości przejść:

$$T(\{s_i\} \to \{s_i'\}) = \begin{cases} exp(-\Delta E / k_B T) & \text{dla } \Delta E > 0, \\ 1 & \text{dla } \Delta E \leqslant 0. \end{cases}$$

WYDZIAŁ FIZYKI UNIWERSYTET WARSZAWSKI

Zastosowanie do modeli sieciowych.

Algorytm Metropolisa dla Modelu Isinga.

1. Wybierz stan początkowy (np. $s_i = 1$ dla $i = 1, 2, \ldots, N$).
2. Wybierz (losowo) węzeł i.
3. Oblicz zmianę energi ΔE gdy s_i zmienia wartość.
4. Wygeneruj liczbę losową r i taką, że $0 < r < 1$.
5. Jeżeli $r < exp(-\Delta E / k_B T)$ to zmianę akceptuj.
6. Idź do 2.

Częstości przejść:

$$T(\{s_i\} \to \{s_i'\}) = \begin{cases} exp(-\Delta E / k_B T) & \text{dla } \Delta E > 0, \\ 1 & \text{dla } \Delta E \leqslant 0. \end{cases}$$

Zastosowanie do modeli sieciowych.

Algorytm Metropolisa dla Modelu Isinga.

1. Wybierz stan początkowy (np. $s_i = 1$ dla $i = 1, 2, \ldots, N$).
2. Wybierz (losowo) węzeł i.
3. Oblicz zmianę energi ΔE gdy s_i zmienia wartość.
4. Wygeneruj liczbę losową r i taką, że $0 < r < 1$.
5. Jeżeli $r < exp(-\Delta E / k_B T)$ to zmianę akceptuj.
6. Idź do 2.

Częstości przejść:

$$T(\{s_i\} \to \{s_i'\}) = \begin{cases} exp(-\Delta E / k_B T) & \text{dla } \Delta E > 0, \\ 1 & \text{dla } \Delta E \leqslant 0. \end{cases}$$

Zastosowanie do modeli sieciowych.

Algorytm Metropolisa dla Modelu Isinga.

1. Wybierz stan początkowy (np. $s_i = 1$ dla $i = 1, 2, \ldots, N$).
2. Wybierz (losowo) węzeł i.
3. Oblicz zmianę energi ΔE gdy s_i zmienia wartość.
4. Wygeneruj liczbę losową r i taką, że $0 < r < 1$.
5. Jeżeli $r < exp(-\Delta E / k_B T)$ to zmianę akceptuj.
6. Idź do 2.

Częstości przejść:

$$T(\{s_i\} \to \{s_i'\}) = \begin{cases} exp(-\Delta E / k_B T) & \text{dla } \Delta E > 0, \\ 1 & \text{dla } \Delta E \leqslant 0. \end{cases}$$

Zastosowanie do modeli sieciowych.

Algorytm Metropolisa dla Modelu Isinga.

1. Wybierz stan początkowy (np. $s_i = 1$ dla $i = 1, 2, \ldots, N$).
2. Wybierz (losowo) węzeł i.
3. Oblicz zmianę energi ΔE gdy s_i zmienia wartość.
4. Wygeneruj liczbę losową r i taką, że $0 < r < 1$.
5. Jeżeli $r < exp(-\Delta E / k_B T)$ to zmianę akceptuj.
6. Idź do 2.

Częstości przejść:

$$T(\{s_i\} \to \{s_i'\}) = \begin{cases} exp(-\Delta E / k_B T) & \text{dla } \Delta E > 0, \\ 1 & \text{dla } \Delta E \leqslant 0. \end{cases}$$

Zastosowanie do modeli sieciowych.

Algorytm Metropolisa dla Modelu Isinga.

1. Wybierz stan początkowy (np. $s_i = 1$ dla $i = 1, 2, \ldots, N$).
2. Wybierz (losowo) węzeł i.
3. Oblicz zmianę energi ΔE gdy s_i zmienia wartość.
4. Wygeneruj liczbę losową r i taką, że $0 < r < 1$.
5. Jeżeli $r < exp(-\Delta E / k_B T)$ to zmianę akceptuj.
6. Idź do 2.

Częstości przejść:

$$T(\{s_i\} \to \{s_i'\}) = \begin{cases} exp(-\Delta E / k_B T) & \text{dla } \Delta E > 0, \\ 1 & \text{dla } \Delta E \leqslant 0. \end{cases}$$

Zastosowanie do modeli sieciowych.

Algorytm Metropolisa dla Modelu Isinga.

1. Wybierz stan początkowy (np. $s_i = 1$ dla $i = 1, 2, \ldots, N$).
2. Wybierz (losowo) węzeł i.
3. Oblicz zmianę energi ΔE gdy s_i zmienia wartość.
4. Wygeneruj liczbę losową r i taką, że $0 < r < 1$.
5. Jeżeli $r < exp(-\Delta E / k_B T)$ to zmianę akceptuj.
6. Idź do 2.

Częstości przejść:

$$T(\{s_i\} \to \{s_i'\}) = \begin{cases} exp(-\Delta E / k_B T) & \text{dla } \Delta E > 0, \\ 1 & \text{dla } \Delta E \leqslant 0. \end{cases}$$

Zastosowanie do modeli sieciowych.

Algorytm Metropolisa dla Modelu Isinga.

1. Wybierz stan początkowy (np. $s_i = 1$ dla $i = 1, 2, \ldots, N$).
2. Wybierz (losowo) węzeł i.
3. Oblicz zmianę energi ΔE gdy s_i zmienia wartość.
4. Wygeneruj liczbę losową r i taką, że $0 < r < 1$.
5. Jeżeli $r < exp(-\Delta E / k_B T)$ to zmianę akceptuj.
6. Idź do 2.

Częstości przejść:

$$T(\{s_i\} \to \{s_i'\}) = \begin{cases} exp(-\Delta E / k_B T) & \text{dla } \Delta E > 0, \\ 1 & \text{dla } \Delta E \leqslant 0. \end{cases}$$

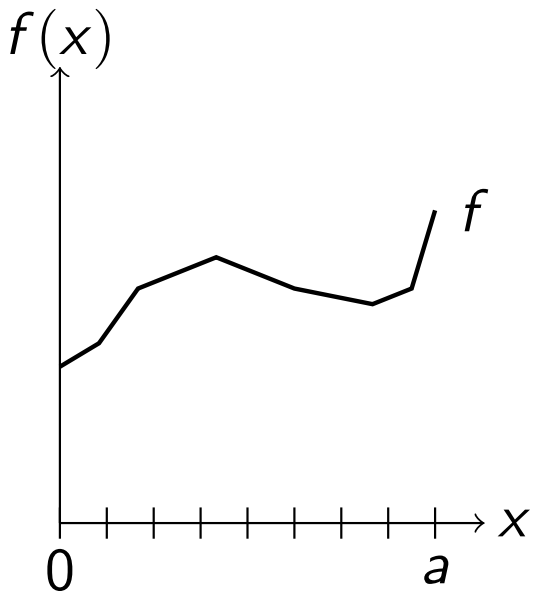
Zastosowanie Algorytmu Metropolisa.

Stochastyczne symulacje komputerowe w:

- fizyce ciała stałego,
- biologii molekularnej,
- chemii.

Plan - przypomnienie.

Dyskretna Transformata Fouriera (DFT).



Problem:

f - funkcja o okresie a, znamy jej N wartości równomiernie rozłożonych na odcinku $[0, a)$

$$f(k\frac{a}{N}) = f_k \quad \text{dla } k = 0, 1, 2, \ldots, N-1.$$

Z N punktów wyznaczamy N współczynników Fouriera c_j ($c_j \to 0$ gdy $j \to \infty$).

Dyskretna Transformata Fouriera (DFT).

Całki w rozkładzie Fouriera:

$$f(x) = \sum_{j=0}^{\infty} c_j e^{\frac{2\pi i}{a} xj},$$

$$c_j = \frac{1}{a} \int_0^a f(x) e^{-\frac{2\pi i}{a} xj} dx.$$

Przybliżone wartości dla N punktów:

$$f_k = \sum_{j=0}^{N-1} c_j^N e^{\frac{2\pi i}{N} jk},$$

$$c_j^N = \frac{1}{N} \sum_{k=0}^{N-1} f_k e^{-\frac{2\pi i}{N} jk}.$$

Def. Dyskretnej Transformaty Fouriera.

Ciąg $c_0^N, c_1^N, \ldots, c_{N-1}^N$.
Koszt $\sim N^2$.

Plan - przypomnienie.

Historia Szybkiej Transformaty Fouriera (FFT).

Wszystko zaczyna sie od Gaussa.

Rozkład: $N = N_1 N_2$, manipulujemy indeksami w $\sum_{j=0}^{N-1}$, $\sum_{k=0}^{N-1}$:

$$\begin{aligned} j = j(a,b) = aN_1 + b, \quad & 0 \leqslant a < N_2, 0 \leqslant b < N_1, \\ k = k(c,d) = cN_2 + d, \quad & 0 \leqslant c < N_1, 0 \leqslant d < N_2. \end{aligned}$$

$$f_{k(c,d)} = \sum_{b=0}^{N_1-1} e^{\frac{2\pi i}{N} b(cN_2+d)} \cdot \sum_{a=0}^{N_2-1} c^N_{j(a,b)} e^{\frac{2\pi i}{N_2} ad}$$

Koszt $\sim (N_1 N_2)(N_1 + N_2)$.

Historia Szybkiej Transformaty Fouriera (FFT).

James W. Cooley i John W. Turkey 1965.

Dane sejsmologiczne, ZSRR, USA $\Rightarrow$ szybki algorytm do analizy sygnałów.
Pomysł: $N = 2^p$ i rekurencyjnie rozkład Gaussa.
Koszt $\sim Nlog(N)$.

John W. Turkey

Zastosowania FFT.

Szybki algorytm do

- analizy spektralnej danych,
- obliczania splotu,
- wykonywania mnożeń dużych liczb, wielomianów, macierzy,
- metod spektralnych w cząstkowych równaniach różniczkowych.

Serce efektywnych algorytmów:

- sortowania,
- transformaty cosinusów (kodowanie MP3),
- kodowania danych (internet, telekomunikacja),
- obróbki obrazu,
- w astronomii (LIGO),
- w kwantowej kryptografii w algorytmie Shor'a.

Podsumowanie

- Temat "Najsłynniejsze Algorytmy XXw." jest małą prowokacją.
- Algorytm Metropolisa jest efektywną metodą obliczeń w rozkładzie kanonicznym.
- Szybka Transformata Fouriera(FFT) "An algoritm the whole family can use." *Daniel N. Rockmore*

Bibliografia I

Theme articles.
The Top 10 Algorithms.
Computing in Science & Engineering, 2(1):22–79, 2000.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall n. Rosenbluth, Augusta H. Teller, and Edward Teller.
Equation of state Calculation by Fast Computing Machines.
The Journal of Chemical Physics, 21(6):1087–1092, 1953.

Plan - przypomnienie.

- Informacje o Twórcach

Metropolis Algoritm for Monte Carlo

1946 von Neumann, Ulam i Metropolis

Stan Ulam, John von Neumann zorientowali się, że statystyczna technika próbkowania zwana odrzucaniem (rejection) nadaje się świetnie do obliczeń dyfuzji neutronów na ENIACu.

Simplex Method for Linear Programming.

1947 Danzig.

Algorytm do rozwiązywania programów liniowych dla planowania i podejmowania decyzji w dużych przedsięwzięciach. Program liniowy zawiera optymalizację funkcji względem więzów będących nierównościami. Sukces tej metody doprowadził do szerokiej gamy uogólnień i specyfikacji do różnych naukowych zastosowań.

Krylov Subspace Iteration Methods.

1950 Hestenes, Stiefel i Lanczos.

Conjugate gradient methods (metody sprzeżonego gradientu) to iterowane algorytmy macierzowe do rozwiązywania bardzo dużych układów równań. Takie układy występują w wielu dziedzinach zastosowań: przepływy płynów, inżynieria mechaniczna, analiza układów półprzewodnikowych, w modelach reakcji jądrowych, w symulacjach obwodów elektrycznych (macierze o milionowych stopniach swobody).

The Decompositoinal approach to Matrix Computations.

1951 Hausholder i Wilkinson.

Rozkład polega na przedstawieniu macierzy jako iloczynu prostszych macierzy.

- LU decomposition,
- QR decomposition,
- singular value decomposition,
- Schur decomposition,
- spectral decomposition,
- eigendecomposition.

Raz uzyskany rozkład staje się platformą, dla której można rozwiązać pewną klasę problemów. Pozwala to przenieść ciężar rozwiązywania problemów na poszukiwanie rozkładów.

The Fortran Optimizing Compiler.

1957 Backus.

John Backus przewodził zespołowi IBM, który pracował nad projektem mającym obniżyć koszty programowania i poszukiwania błedów w programach (debugging). Kompilator stał sie znaczącym czynnikiem umożliwiającym rozwój rozbudowanych systemów oprogramowania.

QR Algorithm for Computing Eigenvalues.

1959-61 Francis.

To algorytm pozwalający obliczać na drodze iteracji wartości własne skomplikowanych macierzy.

Quicksort Algorithm for Sorting.

1962 Hoare.

Problem sortowania, świetnie znany, o wielu zastosowaniach i dużej wadze teoretycznej. Quicksort jest nadal algorytmem najlepszym dla ogólnych danych wejścia.

Fast Fourier Transform.

1965 Cooley i Tukey.

FFT jest jednym z najważniejszych algorytmów stosowanej i obliczeniowej matematyki. Stanowi jądro przetwarzania sygnałów (signal processing). Stosowany w nowoczesnej telekomunikacji.

Integer Relation Detection.

1977 Helaman, Ferguson i Forcade.

Problem znalezienia n liczb naturalnych $a_1, \ldots, a_n \in \mathbb{N}$ (jeśli istnieją) takich, że dla danych $x_1, \ldots, x_n \in \mathbb{N}$ spełniają $a_1 x_1 + \cdots + a_n x_n = 0$. Algorytm rozwiązuje ten problem z zadaną bardzo wysoką dokładnością. Użyto go do odkrycia nie znanych jak dotąd związków algebraicznych oraz do identyfikacji niektórych stałych w kwantowej teori pola, jako kombinacji znanych stałych matematycznych.

Fast Multipole Method.

1987 Greengard i Rokhlin.

Schemat do obliczania sił występujących w grawitacyjnych i elektrycznych problemach N-ciałowych. Schemat prowadzi do zysku rzędu $O(N)$ zamiast $O(N^2)$ jak było w poprzednich algorytmach. Wpłynął na rozwój nowoczesnych N-ciałowych "solwerów" przez wprowadzenie wysoce reguralnego hierarchicznego przestrzennego rozkładu przez ekspansję na różnych poziomach układu.