

GMRES DLA ROZWIĄZYWANIA ŁAŃCUCHÓW MARKOWA NA KOMPUPERZE WEKTOROWYM CRAY SV1

Jarosław Bylina, Beata Bylina

Zakład Informatyki, Instytut Matematyki, Uniwersytet Marii Curie-Skłodowskiej

1. Wstęp

Łańcuchy Markowa (będące klasą procesów stochastycznych) są szeroko stosowanym narzędziem do modelowania zachowań różnego rodzaju układów dynamicznych. W połączeniu z modelami kolejkowymi świetnie nadają się do badania sieci komunikacyjnych i komputerowych.

Żeby reprezentować zachowanie takiej sieci (lub innego obiektu) za pomocą łańcucha Markowa, należy wygenerować przestrzeń wszystkich możliwych stanów modelu (i ponumerować je kolejnymi liczbami naturalnymi od 1 do n), a następnie wyznaczyć wszystkie potencjalne przejścia pomiędzy stanami oraz intensywności $q_{ij}(t)$ przejść. W ten sposób powstaje macierz $\mathbf{Q}(t)$, opisująca całkowicie i jednoznacznie zachowanie reprezentowanego łańcuchem Markowa obiektu w czasie.

Elementy macierzy $\mathbf{Q}(t)$ określone przez równości:

$$q_{ij}(t) = \lim_{\Delta t \rightarrow 0} \frac{p_{ij}(t, t + \Delta t)}{\Delta t}, \quad \text{dla } i \neq j, \quad (1)$$

$$q_{ii}(t) = -\sum_{j \neq i} q_{ij}(t), \quad (2)$$

gdzie $p_{ij}(t_1, t_2)$ jest prawdopodobieństwem, że układ w chwili t_2 znajdzie się w stanie j pod warunkiem, że w chwili t_1 był w stanie i .

W interesujących nas przypadkach mamy do czynienia z jednorodnym łańcuchem Markowa, to jest w takim, w którym sama macierz $\mathbf{Q}(t)$ nie zmienia się w czasie – stąd dalsze oznaczenie \mathbf{Q} .

W badaniach, w których używamy łańcuchów Markowa najbardziej interesującym jest wyznaczenie rozkładu prawdopodobieństw stanów w danej chwili [6]. Rozkład taki reprezentowany jest przez wektor $\boldsymbol{\pi}(t) = (\pi_1, \dots, \pi_n)$ prawdopodobieństw poszczególnych stanów, który spełniać musi pewne dodatkowe – oczywiste – ograniczenia:

$$\boldsymbol{\pi}(t) \geq 0, \quad \|\boldsymbol{\pi}(t)\|_1 = 1. \quad (3)$$

Wektor $\boldsymbol{\pi}(t)$ związany jest z macierzą \mathbf{Q} następującą zależnością:

$$\frac{d\boldsymbol{\pi}(t)}{dt} = \boldsymbol{\pi}(t)\mathbf{Q}. \quad (4)$$

W przypadku, gdy chcemy wyznaczyć prawdopodobieństwa stacjonarne (niezależne od czasu), zależność (4) redukuje się do

$$0 = \boldsymbol{\pi}\mathbf{Q}. \quad (5)$$

Po podstawieniu (dla wygody) $\mathbf{x} = \boldsymbol{\pi}^T$ do (5) oraz po uwzględnieniu warunków (3) dostajemy do rozwiązania problem:

$$\mathbf{Q}^T \mathbf{x} = 0, \quad \mathbf{x} \geq 0, \quad \|\mathbf{x}\|_1 = 1. \quad (6)$$

Problem ten ma co najmniej jedno rozwiązanie, ponieważ $\text{rank } \mathbf{Q} \leq n-1$ (co bezpośrednio wynika z (2)). Co więcej, w przypadku, gdy $\text{rank } \mathbf{Q} = n-1$ takie rozwiązanie (spełniające (3)) jest dokładnie jedno.

2. Iterowany algorytm GMRES

Istnieje wiele metod rozwiązywania równań (6). Mianowicie, mamy metody bezpośrednie (zwane też „dokładnymi” – jak choćby eliminacja Gaussa), metody iteracyjne (Jacobiego, Gaussa-Seidla, SOR), metody projekcyjne i inne. Jedną z bardziej interesujących i skutecznych jest iterowana wersja algorytmu GMRES (metoda projekcyjna).

Algorytm GMRES [5] rozpoczyna się znalezieniem znormalizowanej reszty \mathbf{v}_1 obliczonej dla początkowego przybliżenia \mathbf{x}_0 :

$$\mathbf{r}_0 = -\mathbf{Q}^T \mathbf{x}_0, \quad (7)$$

$$\beta = \|\mathbf{r}_0\|_2, \quad (8)$$

$$\mathbf{v}_1 = \frac{\mathbf{r}_0}{\beta}. \quad (9)$$

Następnie konstruowana jest ortonormalna baza (\mathbf{v}_j) dla podprzestrzeni Kryłowa o wymiarze m (przy czym $m \ll n$) poprzez proces Arnoldiego (dla $j = 1, \dots, m$):

$$\mathbf{w} = \mathbf{Q}^T \mathbf{v}_j, \quad (10)$$

$$\text{dla } i = 1, \dots, j: \quad h_{ij} = \mathbf{v}_i^T \mathbf{w}, \quad \mathbf{w} = \mathbf{w} - h_{ij} \mathbf{v}_i, \quad (11)$$

$$h_{j+1,j} = \|\mathbf{w}\|_2, \quad (12)$$

$$\mathbf{v}_{j+1} = \frac{\mathbf{w}}{h_{j+1,j}}. \quad (13)$$

Kolejnym krokiem jest znalezienie (dzięki metodzie najmniejszych kwadratów) wektora $\mathbf{y} = (y_1, \dots, y_m)^T$ minimalizującego wyrażenie

$$\|\beta \mathbf{e}_1 - \mathbf{H}\mathbf{y}\|_2, \quad (14)$$

gdzie wektor $\mathbf{e}_1 = (1, 0, \dots, 0)^T$ jest wymiaru $m+1$, a macierz \mathbf{H} jest wymiaru $(m+1) \times m$ oraz:

$$\mathbf{H} = \begin{cases} h_{ij}, & \text{dla } i \leq j+1, \\ 0, & \text{dla } i > j+1. \end{cases} \quad (15)$$

W iterowanym algorytmie GMRES, jeżeli wynikowy wektor

$$\mathbf{x} = \mathbf{x}_0 + \sum_{i=1}^m \mathbf{v}_i y_i \quad (16)$$

daje zbyt dużą (w sensie wybranej normy) resztę $\mathbf{r} = -\mathbf{Q}^T \mathbf{x}$ (a liczba powtórzeń nie przekroczyła jeszcze maksymalnej), to wszystkie operacje powtarzane są od początku, ale już dla $\mathbf{x}_0 = \mathbf{x}$.

3. Wektorowy iterowany algorytm GMRES

Jedną z zalet algorytmu GMRES (oraz jego wersji pochodnych – w tym iterowanej) jest duża liczba operacji wektorowych, które mogą być zaimplementowane wprost jako operacje wektorowe – a nie sekwencje operacji skalarnych (taka budowa algorytmu pozwala także na jego skuteczne zrównoleglenie [1]). Owe operacje wektorowe to: obliczanie drugiej normy wektora – w (8), (12); skalowanie wektora – w (9), (13); podstawianie wektora – w (9), (13), (16); iloczyn skalarny wektorów – w (11); oraz sumowanie wektorów ze skalowaniem jednego z nich – w (11), (16).

Operacją wektorową – która jednak w naszym przypadku musi być zaimplementowana skalarnie – jest też mnożenie macierzy \mathbf{Q} przez wektor w równościach (7) oraz (10). Niestety, macierz \mathbf{Q} , jako macierz rzadka, nie jest przechowywana w standardowej formie, lecz w formacie Harwella-Boeinga, więc operacje z jej udziałem nie nadają się wprost do implementacji wektorowej.

Opisanym powyżej operacjom na wektorach odpowiadają bezpośrednio podprogramy pierwszego poziomu biblioteki (a właściwie standardu) *BLAS* (*Basic Linear Algebra Subprograms*, [4]) – dostępnej na wiele platform w różnorodnych implementacjach. Obliczanie drugiej normy wektora realizuje podprogram **?NRM2**, skalowanie wektora – podprogram **?SCAL**, podstawianie wektora pod wektor (kopiowanie) – podprogram **?COPY**, iloczyn skalarny wektorów – podprogram **?DOT**, a sumowanie wektorów ze skalowaniem jednego z nich ($\mathbf{y} \leftarrow \alpha \mathbf{x} + \mathbf{y}$) – podprogram **?AXPY** (we wszystkich nazwach podprogramów zamiast ? znajduje się jedna z liter **S**, **D**, **C**, **Z** oznaczających typ składowych wektora – odpowiednio: rzeczywiste

pojedynczej precyzji, rzeczywiste podwójnej precyzji, zespolone pojedynczej precyzji i zespolone podwójnej precyzji).

4. System *Cray SVI*

Doświadczenia przeprowadzone zostały na komputerze *Cray SVIex-1* [3], działającym pod systemem operacyjnym *UNICOS* w wersji 10.0.1.1 opartym na *USL System V*. Wykorzystany tu typ komputera charakteryzuje się posiadaniem wielu procesorów (wektorowych), które – pogrupowane po 4 – tworzą tak zwane *MSP (Multi Streaming Processor)* – procesory wielostrumieniowe, których działanie przypomina z jednej strony działanie pojedynczych procesorów wektorowych, a z drugiej – układów *SMP*. Cała czwórka przydzielana jest bowiem jednocześnie do jednego zadania – w ten sposób (w odróżnieniu od tradycyjnej równoległości) część czasu procesorów może zostać zmarnowana, ale z drugiej strony cała czwórka jest zarządzana i synchronizowana wspólnie (a przez to wydajniej), co pozwala na wykorzystanie równoległości na niższym poziomie.

Inną ważną cechą użytej maszyny jest brak pamięci wirtualnej – cała pamięć operacyjna to pamięć rzeczywista (32 GB RAM). W odniesieniu do prezentowanego przez nas wykorzystania maszyny (w testach największa z macierzy \mathbf{Q} zajmowała ponad 1GB pamięci) jest to znaczące udogodnienie, ponieważ nie zachodzi niebezpieczeństwo błędów stronicowania (*page faults*) i związanych z nimi opóźnień w wykonywaniu algorytmu.

5. Doświadczenia

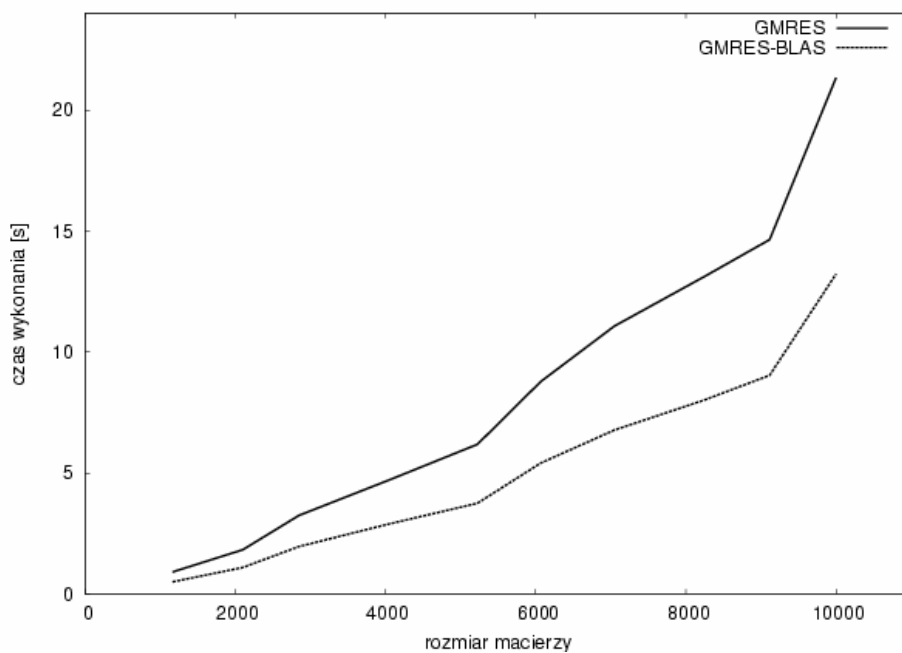
Opisany w sekcji 2 iterowany algorytm GMRES został zaprogramowany w języku *C* przy użyciu pojedynczej precyzji (na opisywanym komputerze oznacza to liczbę ośmiobajtową) i skompilowany przy pomocy standardowego kompilatora dostępnego na opisaną wyżej maszynę, to jest za pomocą *Cray Standard C* w wersji 6.5.0.5. Przy kompilacji włączone były także opcje automatycznej optymalizacji najwyższego poziomu.

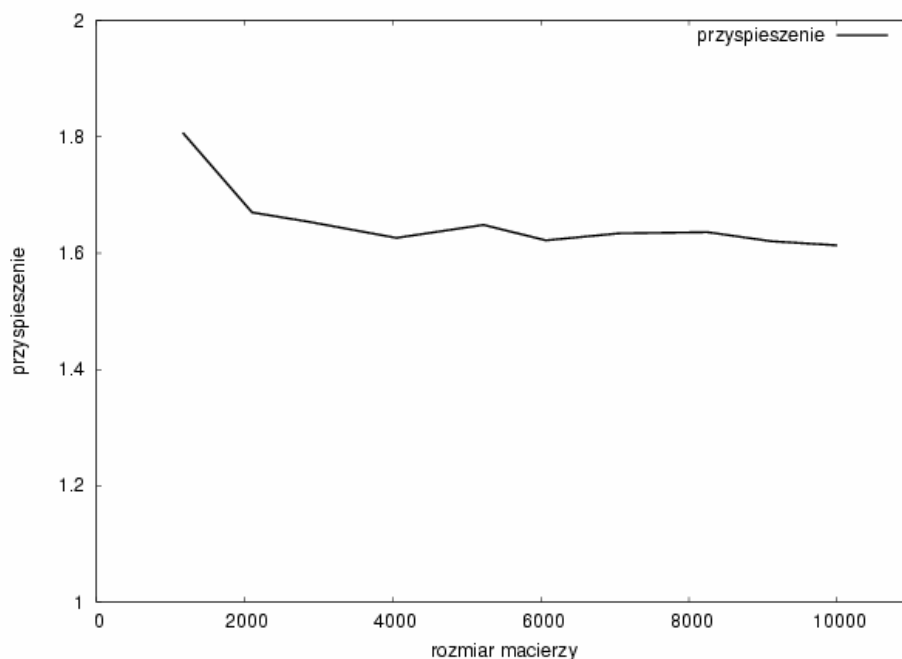
Tenże algorytm w wersji wektorowej został zaprogramowany także w języku *C* (i skompilowany tym samym kompilatorem, także w pojedynczej precyzji), ale z użyciem standardowo dostępnej biblioteki *BLAS*. Wszystkie operacje wektorowe zostały zastąpione odpowiednimi operacjami powyższej biblioteki zgodnie z opisem w sekcji 3.

Obie wersje przetestowano z tymi samymi parametrami (wymiar przestrzeni Kryłowa: $m = 35$; warunek zakończenia obliczeń: $\|-\mathbf{Q}^T \mathbf{x}\|_2 < 10^{-9}$; maksymalna liczba iteracji: 1000) oraz tymi samymi zestawami danych. Jako dane testowe służyły macierze przejścia dla Markowowskiego modelu zachowania się ciekącego wiadra z różnymi parametrami [2]. Parametry danych oraz wyniki testów przedstawiono poniżej, w tabeli i na wykresach. Widać z nich, że ręczna wektoryzacja przy pomocy biblioteki *BLAS* (już na poziomie pierwszym) może znacząco poprawić wydajność – nawet w przypadku kompilatorów, mających wbudowaną możliwość automatycznej

wektoryzacji, która powinna – zdawałoby się – działać wystarczająco dobrze dla tak prostego i wektorowego z natury algorytmu jak GMRES.

wymiar macierzy n	liczba elementów niezerowych nz	średnio niezerowych w wierszu $\frac{nz}{n}$	liczba iteracji	czas działania algorytmu bez modyfikacji t_1 [s]	czas działania algorytmu z biblioteką BLAS t_2 [s]	przyspieszenie $\frac{t_1}{t_2}$
1166	7387	6,34	8	0,912353	0,504895	1,807015
2101	14348	6,83	8	1,833199	1,097697	1,670041
2850	19977	7,01	10	3,254391	1,967248	1,654286
4049	29198	7,21	10	4,714422	2,899082	1,626178
5225	38538	7,38	10	6,189704	3,754537	1,648593
6069	44850	7,39	12	8,786772	5,417535	1,621913
7049	52018	7,38	13	11,078186	6,780063	1,633936
8249	61920	7,51	13	13,128827	8,024775	1,636037
9116	69791	7,66	13	14,653699	9,043371	1,620380
10001	75394	7,54	17	21,350315	13,232359	1,613493





6. Podziękowania

W pracy wykorzystano wyniki uzyskane z wykorzystaniem zasobów komputerowych Interdyscyplinarnego Centrum Modelowania Matematycznego i Komputerowego (ICM) Uniwersytetu Warszawskiego.

Literatura

- [1] J. Bylina: Distributed solving of Markov chains for computer network models, *Annales UMCS Informatica*, 1 (2003), Lublin 2003, str. 15–20.
- [2] J. Domańska, T. Czachórski: Wpływ samopodobnej natury ruchu na zachowanie mechanizmu ciekącego wiadra, *Studia Informatica*, Vol. 24 (2003), nr 2A (53), Gliwice 2003, str. 199–212.
- [3] <http://www.cray.com/products/systems/sv1/>
- [4] <http://www.netlib.org/blas/index.html>
- [5] Y. Saad, M.H. Schultz: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.*, 7 (3) (1986), str. 856–869.
- [6] W.J. Stewart: *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ 1994.