ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: INFORMATYKA z. 30 Nr kol. 1315

Joanna DOMAŃSKA
Mirosław NIEMIEC
Piotr PECKA
Joanna TOMASIK

NUMERYCZNE PROBLEMY WYKORZYSTANIA MARKOWOWSKICH MODELI SIECI KOMPUTEROWYCH

Streszczenie. Artykuł przedstawia algorytmy tworzenia macierzy tranzycji w markowowskich modelach systemów komputerowych oraz numeryczne sposoby rozwiązywania modeli systemów o bardzo dużej liczbie stanów. Użyteczność algorytmów zilustrowana została przykładami numerycznymi.

NUMERICAL PROBLEMS OF UTILISATION OF MARKOV MODELS OF COMPUTER NETWORKS

Summary. The article presents algorithms for finding transition matrices in Markov models of computer systems as well as numerical solutions for models with a very big number of states. The usability of the algorithms is demonstrated by numerical examples.

LES PROBLEMES NUMERIQUES D'EXPLOITATION DES MODELS DE MARKOV DANS LES RESEAUX D'ORDINATEURS

Résumé. L'article presente plusieurs algoritmes permettant la création des matrices de transition des systèmes d'ordinateurs suivant le model de Markov, anisi que les manières de resolutions numeriques de tels models contenant un grand nombre d'états. L'utilité des approches présentés est testée par des exemples numeriques.

Niniejszy tekst został opracowany częściowo w ramach Projektu Badawczego KBN nr 8 T11C 032 08.

1. Wstęp

Modelowanie za pomocą *łańcuchów Markowa* stosowane jest w wielu dziedzinach nauki i techniki. W informatyce używa się modeli wykorzystujących łańcuchy Markowa między innymi do lokalizacji zatłoczeń w sieciach komputerowych, do oszacowania korzyści wynikających ze zwiększania liczby procesorów w systemach wieloprocesorowych, do określania wpływu wyboru algorytmu szeregowania na przepustowość.

Łańcuchy Markowa są dyskretnym procesem stochastycznym posiadającym własność braku pamięci: prawdopodobieństwa przejść między stanami zależą tylko od tych stanów, a nie zależą od historii procesu.

Niech łańcuch Markowa z ciągłym czasem ma następującą przestrzeń stanów: $\{x_1, x_2, ..., x_n\}$. Jeżeli $X(t) = x_i$, mówimy, że łańcuch jest w stanie i. Oznaczmy:

$$P_i(t) = P[X(t) = x_i] \quad \text{oraz} \quad p_{ij}(s,u) = P[X(u) = x_j | X(s) = x_i]$$

Dla jednorodnego łańcucha Markowa

$$p_{ij}(t, t+\Delta t) = p_{ij}(\Delta t) = q_{ij}\Delta t$$

Współczynnik q_{ij} nazywamy współczynnikiem (intensywnością) przejścia (tranzycji) ze stanu i do stanu j.

Uzyskanie rozwiązania polega na określeniu prawdopodobieństw stanów $P_i(t)$ poprzez rozwiązanie układu równań różniczkowych:

$$\frac{dP_i(t)}{dt} = \sum_j P_j(t) q_{ji} \tag{1}$$

lub w zapisie macierzowym:

$$\frac{dp(t)}{dt} = Q^T p(t) \tag{2}$$

p jest poszukiwanym wektorem prawdopodobieństw stanów, a macierz tranzycji Q jest nazywana *generatorem infinitezymalnym* prawdopodobieństw stanów. Układ równań (1) należy uzupełnić warunkiem normalizacyjnym $\sum_j p_j(t) = 1$.

W stanie ustalonym, gdy prawdopodobieństwa stanów nie zależą od czasu, $p(t) = p$, układ (2) przechodzi w

$$0 = Q^T p \tag{3}$$

Wymiar Q, p zależy od liczby stanów przyjmowanych przez proces.

Modelowanie danego systemu za pomocą łańcuchów Markowa sprowadza się więc do znalezienia macierzy tranzycji (takiej samej dla rozwiązania ustalonego i nieustalonego) i rozwiązania układu równań (2) lub (3) - w zależności od tego, czy poszukujemy rozwiązania w stanie nieustalonym, czy ustalonym.

W ogólnym przypadku rozwiązania poszukujemy na drodze numerycznej, a problemy, jakie napotykamy, związane są z bardzo dużą, sięgającą setek tysięcy, liczbą stanów (a więc wymiarem macierzy Q). Macierz Q jest z reguły rzadka i źle uwarunkowana,

co potęguje trudności i wyklucza stosowanie najbardziej popularnych algorytmów rozwiązywania układów równań algebraicznych i różniczkowych.

W niniejszym artykule opisano metody generowania macierzy tranzycji Q oraz przedstawiono zastosowanie metody podprzestrzeni Kryłowa, jako najbardziej odpowiedniej dla tego typu zagadnień, do rozwiązywania stanów ustalonych i nieustalonych w markowowskich modelach systemów komputerowych. Użyteczność algorytmów zilustrowana została odpowiednimi przykładami numerycznymi.

2. Generowanie macierzy tranzycji

Transponowana macierz tranzycji Q^T zawiera współczynniki układu równań (1). N–ty wiersz tej macierzy zawiera współczynniki n–tego równania opisującego bilans intensywności przejść dla n–tego stanu sieci. Wynika stąd, że jeżeli w n–tym wierszu m–ty element jest niezerowy, to stan m–ty jest stanem poprzednim względem stanu n–tego, a wartość elementu $[n, m]$ jest wartością współczynnika intensywności q_{mn} przejścia ze stanu m do n. Element q_{nn} stojący na przekątnej macierzy jest wziętą ze znakiem minus sumą współczynników intensywności przejść do wszystkich stanów następnych względem stanu n–tego.

Algorytm generowania n–tego wiersza macierzy tranzycji Q polega na:

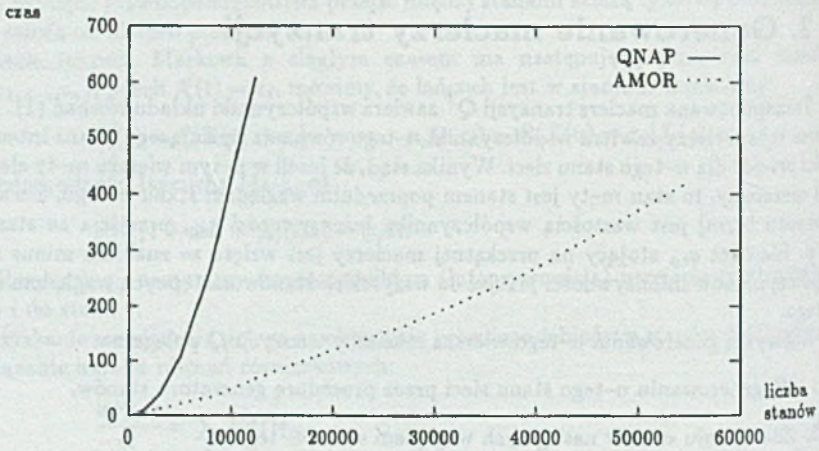
1. Wygenerowaniu n–tego stanu sieci przez procedurę generatora stanów.
2. Znalezieniu stanów następnych względem stanu n–tego.
3. Obliczeniu numerów porządkowych stanów następnych, aby wiedzieć, na jakich pozycjach w n–tym wierszu umieścić współczynniki przejść ze stanu n–tego do jego stanów następnych.
4. Obliczeniu współczynników przejść i umieszczeniu ich na odpowiednich pozycjach w wierszu.
5. Zsumowaniu obliczonych współczynników przejść i umieszczeniu ich ze znakim minus w należącym do przekątnej macierzy elemencie wiersza.

Aby można było taką macierz zbudować, należy znaleźć metodę numeracji stanów. Algorytm numeracji stanów i budowy macierzy tranzycji dla modeli markowowskich opisujących sieci stanowisk masowej obsługi zaproponował Stewart i opisał w [10] i [11]. Metoda ta jest wykorzystana w modelach markowowskich najpopularniejszego chyba w Europie pakietu oceny efektywności systemów komputerowych QNAP [13]. Duża zajętość pamięci komputera oraz konieczność wielokrotnego przeszukiwania listy stanów powoduje, że metoda Stewarta jest nieefektywna w przypadku modeli o dużej liczbie stanów zarówno pod względem czasowym, jak i pamięciowym.

Metoda numeracji stanów, alternatywna w stosunku do metody Stewarta, opiera się na idei generowania kolejnych stanów sieci przez procedurę generatora stanów. Generator stanów określa pewne, wzajemnie jednoznaczne odwzorowanie zbioru liczb naturalnych

wraz z zerem na zbiór stanów sieci. Musi więc istnieć funkcja odwrotna, która na podstawie wektora stanu sieci oblicza jego numer. Zaproponowana przez autorów niniejszego artykułu metoda numeracji stanów [4] (zwana metodą funkcji odwrotnej) polega na znalezieniu analitycznej postaci funkcji odwrotnej i obliczaniu na jej podstawie numerów stanów. Algorytm wykorzystujący opisaną metodę został zaimplementowany w pakiecie programowym AMOR [1] modelującym działanie systemów komputerowych.

Zaletą metody funkcji odwrotnej jest niewielka zajętość pamięci oraz bardzo krótki czas generacji macierzy. Rysunek 1 przedstawia wykres zależności czasu obliczania pa-



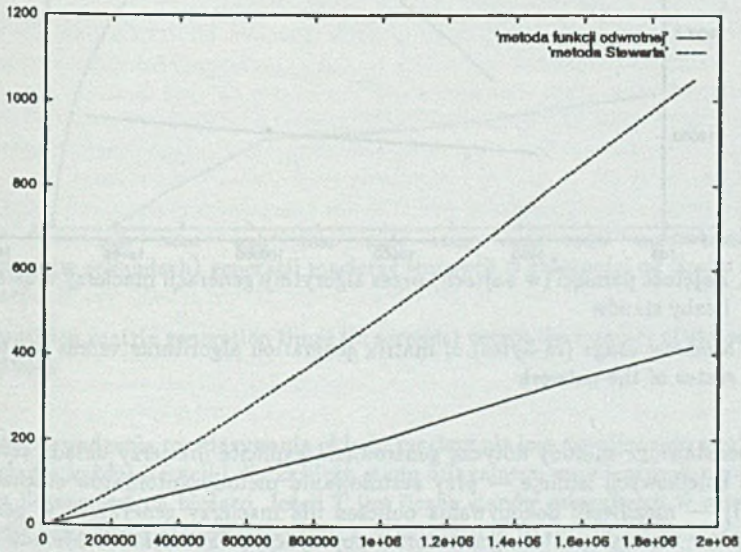
Rys. 1. Czasy (w sekundach) obliczania parametrów sieci, w zależności od liczby stanów, dla pakietów QNAP i AMOR

Fig. 1. Network parameters computation times (in seconds) versus the number of states for QNAP and AMOR software kits

rametrów sieci (czas generacji układu równań + czas rozwiązywania układu) od liczby jej stanów, odpowiednio dla pakietów QNAP i AMOR. Obliczenia przeprowadzono na komputerze SUN SparcStation5 z 32 MB pamięci RAM. W przypadku pakietu QNAP przekroczenie 12 tysięcy stanów stało się niemożliwe z powodu niewystarczającej ilości pamięci. Pakiet AMOR (na tym samym sprzęcie) pozwalał na obliczanie sieci mających do 180 tysiecy stanów, a na komputerze wektorowym Convex C3820 osiągnięto milion stanów i nie była to jeszcze granica możliwości tego pakietu.

Wadą metody funkcji odwrotnej jest to, że za jej pomocą można rozwiązywać tylko takie sieci, dla których zbiór stanów nie zależy od ich topologii. Idea generatora stanów opiera się bowiem na mechanizmie przerzucania klientów i generowania stanów wewnętrznych dla poszczególnych elementów sieci. Należy zauważyć, że mechanizm przerzucania klientów abstrahuje od topologii sieci, czyli opiera się na założeniu, że każdy klient może znaleźć się na dowolnym stanowisku sieci – przy wszystkich możliwych kombinacjach rozłożeń pozostałych klientów w sieci. Powyższe założenie zawęża zbiór sieci rozwiązy-

walnych za pomocą generatora, wyłączając z niego sieci, dla których istnieją nietypowe algorytmy wyboru drogi. Do rozwiązywania takich przypadków należy użyć algorytmu opartego na idei Stewarta, w którym można jedynie zmodyfikować sposób reprezentacji listy stanów w pamięci oraz sposób przeszukiwań tych stanów – opierając się na idei struktur drzewiastych. Zastosowanie powyższego algorytmu jest jednak związane z wydłużeniem czasu generowania macierzy i zwiększeniem rozmiaru potrzebnej pamięci. Rysunki 2 i 3 przedstawiają zależności czasu generacji macierzy i zajętości pamięci od liczby stanów dla metody funkcji odwrotnej i metody Stewarta, w której zastosowano strukturę drzewiastą.

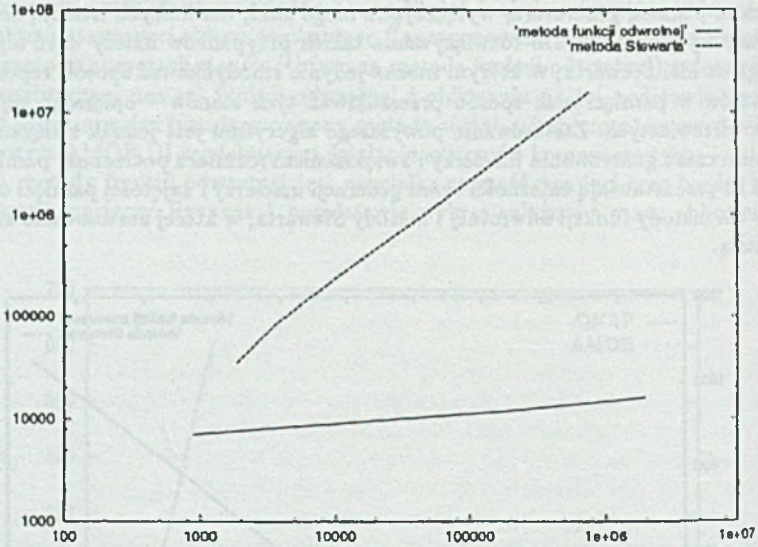


Rys. 2. Czasy (w sekundach) generacji macierzy tranzycji w zależności od liczby stanów sieci

Fig. 2. Transition matrix generation times (in seconds) versus the number of states of the network

Metoda funkcji odwrotnej pozwala na bardzo szybkie, w stosunku do metody Stewarta, generowanie macierzy tranzycji. Dodatkową jej zaletą jest bardzo mała zajętość pamięci. Należy tutaj podkreślić, że rys. 2 i 3 przedstawiają porównanie metody funkcji odwrotnej z metodą Stewarta, w której zastosowano zmieniony sposób reprezentacji w pamięci listy stanów i metodę przeszukiwania tych stanów, opartą na idei drzewa. Rysunek 4 przedstawia porównanie czasów generacji macierzy dla metody funkcji odwrotnej i „oryginalnej" metody Stewarta, czyli takiej, jaką przedstawiono w [10] i [11].

Wziąwszy pod uwagę opisane ograniczenia metody funkcji odwrotnej, optymalnym rozwiązaniem byłoby znalezienie metody łączącej w sobie metodę funkcji odwrotnej i metodę Stewarta. System rozpoznawałby, które części sieci należy rozwiązywać pierwszą metodą, a które drugą metodą.



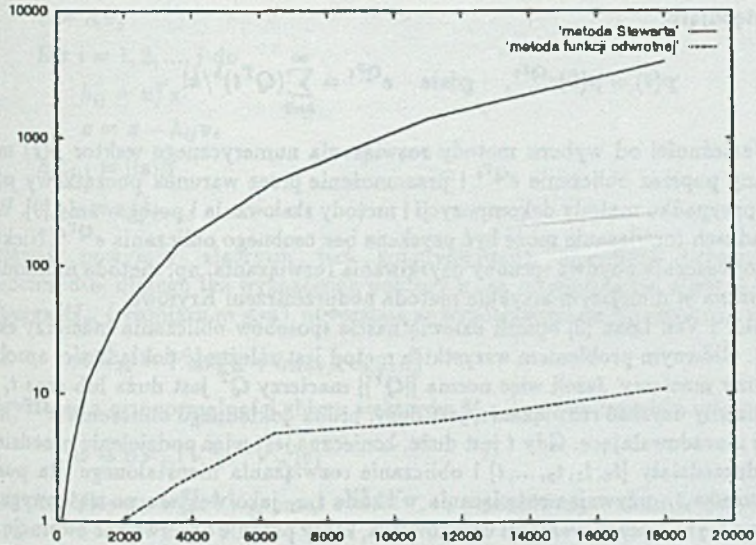
Rys. 3. Zajętość pamięci (w bajtach) przez algorytmy generacji macierzy w zależności od liczby stanów

Fig. 3. Memory usage (in bytes) of matrix generation algorithms versus the number of states of the network

Przedstawione metody dotyczą generowania explicite macierzy układu równań. Dla modeli kolejkowych istnieje — przy zastosowaniu metody automatów stochastycznych [5, 6, 7] — możliwość dokonywania obliczeń dla macierzy generatora w postaci sum i iloczynów tensorowych dla generatorów pojedynczych automatów. Metoda może być zastosowana w sieciach, w których występują elementy synchronizacji i równoległości, a więc w sieciach bardzo złożonych z punktu widzenia metod bezpośredniego budowania całego generatora systemu i polega na tym, aby wyodrębnić z modelu jednowymiarowe procesy Markowa, które składają się na wielowymiarowy proces opisujący działanie sieci, wyznaczenie dla tych procesów odpowiadających im generatorów i wyznaczenie zależności między pojedynczymi procesami, które mają charakter wymuszeń synchronizacyjnych. Dla ciągłych procesów Markowa generator globalny ma postać:

$$Q = \bigoplus_{i=1}^{N} Q^{(i)} + \sum_{j=1}^{K} \Big(\bigotimes_{i=1}^{N} S_j^{(i)} - \bigotimes_{i=1}^{N} R_j^{(i)} \Big), \tag{4}$$

przy czym w sieci znajduje się N automatów stochastycznych i występuje K zdarzeń synchronizacyjnych. Macierze $Q^{(i)}$ to generatory dla poszczególnych automatów, macierze $S_j^{(i)}$ to macierze tranzycji dla poszczególnych automatów i zdarzeń synchronizujących, a macierze $R_j^{(i)}$ to macierze normalizujące takie, że $R_j^{(i)}$ jest macierzą diagonalną i $\bigwedge_l \Big(\sum_k (S_j^{(i)})_{lk} - (R_j^{(i)})_{ll} = 0 \Big)$.



Rys. 4. Czasy (w sekundach) generacji macierzy tranzycji w zależności od liczby stanów sieci

Fig. 4. Transition matrix generation times (in seconds) versus the number of states of the network

Podczas prowadzenia rozwiązywania układu macierz nie jest rozwijana do swojej pełnej postaci, dla każdej iteracji i dla każdego stanu osiągalnego współczynnik z macierzy generatora liczony jest na bieżąco. Jeżeli T jest liczbą stanów osiągalnych w całym systemie, a $d^{(i)}$ jest liczbą stanów osiągalnych w automacie i-tym, to na każdym kroku iteracyjnym mamy łącznie (por. [12]):

$$NT\left(\frac{K}{2}+\sqrt{\sum_{i=1}^{N}(d^{(i)}-1)/\prod_{i=1}^{N}d^{(i)}}\right)$$

operacji dodawania i mnożenia.

3. Rozwiązywanie układu równań różniczkowych zwyczajnych metodą podprzestrzeni Kryłowa

Jeżeli macierz tranzycji Q została znaleziona, to poszukiwany wektor prawdopodobieństw stanów $p(t)$ dla konkretnej chwili czasowej otrzymamy po rozwiązaniu układu równań różniczkowych zwyczajnych (2). Analityczne rozwiązanie takiego układu wyraża

się następująco:

$$p(t) = p(0)e^{Q^T t}, \quad \text{gdzie} \quad e^{Q^T t} = \sum_{k=0}^{\infty} (Q^T t)^k / k!$$

W zależności od wyboru metody rozwiązania numerycznego wektor $p(t)$ może być uzyskany poprzez obliczenie $e^{Q^T t}$ i przemnożenie przez warunek początkowy $p(0)$. Tak jest w przypadku metody dekompozycji i metody skalowania i potęgowania [9]. W innych przypadkach rozwiązanie może być uzyskane bez osobnego obliczania $e^{Q^T t}$. Niektóre metody dopuszczają obydwa sposoby uzyskiwania rozwiązania, np. metoda uniformizacji [9] czy opisana w niniejszym artykule metoda podprzestrzeni Krylowa.

Moler i Van Loan [3] opisali dziewiętnaście sposobów obliczania macierzy eksponencjalnej. Głównym problemem wszystkich metod jest zależność dokładności aproksymacji od normy macierzy. Jeżeli więc norma $||Q^T||$ macierzy Q^T jest duża lub czas t, dla którego chcemy uzyskać rozwiązanie, jest duży, próba dokładnego obliczenia $e^{Q^T t}$ może dać wyniki niezadowalające. Gdy t jest duże, konieczne jest więc podzielenie przedziału $[t_0, t]$ na podprzedziały $[t_0, t_1, t_2, ..., t]$ i obliczanie rozwiązania nieustalonego dla poszczególnych czasów t_j, używając rozwiązania w czasie t_{j-1} jako wektora początkowego. Często pokrywa się to z oczekiwaniami użytkownika, który pragnie obserwować ewolucję systemu w czasie.

Większość metod używanych do rozwiązywania układu równań (2) nie sprawdza się, jeżeli obie wielkości: $||Q^T||$ i t są duże. Trudno jest więc wskazać metodę odpowiednią dla każdego przypadku. Jedynie stosunkowo młoda metoda podprzestrzeni Kryłowa radzi sobie z dużymi rozmiarami powyższych wielkości, dlatego też jest ona szczególnie polecana dla tego typu zagadnień [8], [9].

Dla ułatwienia zapisu przyjmijmy następujące oznaczenia: $A = Q^T$, $v = p(t_i)$, a $w = p(t_{i+1})$. Poszukiwanym rozwiązaniem jest więc $w = e^A v$ (dla ułatwienia zapisu można opuścić na razie stałą $\tau_i = t_{i+1} - t_i$).

Podprzestrzeń Kryłowa jest rozpięta na wektorach $v, Av, ..., A^{m-1}v$:

$$K_m(A, v) \equiv span\{v, Av, ..., A^{m-1}v\}$$

Opisywana metoda sprowadza się do znalezienia takiego elementu tej podprzestrzeni, który najlepiej przybliża poszukiwany wektor w. Zbiór wektorów bazowych tej podprzestrzeni oznaczmy przez $V_m = [v_1, v_2, ..., v_m]$. Jeżeli przyjmiemy $v_1 = v/\beta$, gdzie $\beta = ||v||_2$, wtedy $v = \beta V_m e_1$ i można zapisać:

$$\begin{aligned} w \approx V_m V_m^{-1} V_m^{-T} V_m^T e^A v &= V_m[(V_m^T V_m)^{-1} V_m^T e^A v] \\ &= V_m[(V_m^T V_m)^{-1} V_m^T e^A V_m]\beta e_1 \end{aligned} \tag{5}$$

Wektor e_i jest wektorem, który ma i-ty element równy jeden, a pozostałe elementy równe zero.

Zbiór wektorów bazowych V_m wyznaczamy korzystając z tzw. *procedury Arnoldiego*:

1. $v_1 = v/||v||_2$

2. For $j = 1, 2, ..., m$ do

$z = Av_j$

For $i = 1, 2, ..., j$ do

$h_{ij} = v_i^T z$

$z = z - h_{ij} v_i$

$h_{j+1,j} = ||z||_2$

$v_{j+1} = z/h_{j+1,j}$

Ponieważ powyższy algorytm jest zmodyfikowaną procedurą ortogonalizacji Grama-Schmidta, dlatego też wyznaczone wektory v_j są ortonormalne, a górna macierz Hessenberga H_m (rozmiaru $m \times m$), utworzona ze współczynników h_{ij}, spełnia zależność:

$$AH_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T \tag{6}$$

Korzystając z ortonormalności zbioru wektorów V_m możemy uprościć zależność (5):

$$w \approx \beta V_m (V_m^T e^A V_m) e_1$$

Jeżeli przybliżymy $V_m^T e^A V_m$ przez $e^{V_m^T A V_m}$, to poszukiwany wektor w przyjmie postać:

$$w \approx \beta V_m e^{V_m^T A V_m} e_1$$

Korzystając z ortonormalności zbioru wektorów V_m, możemy przekształcić zależność (6):

$$H_m = V_m^T A V_m \tag{7}$$

Poszukiwane rozwiązanie przyjmie więc postać:

$$w \approx \beta V_m e^{H_m} e_1$$

Uzyskane rozwiązanie nadal wymaga obliczania macierzy eksponencjalnej, ale małego rozmiaru (m – rozmiar podprzestrzeni Kryłowa jest dużo mniejszy od n – liczby stanów badanego systemu - w stosowanym algorytmie autorzy przyjęli m rzędu dziesięciu). Do jej obliczenia możemy użyć dowolnej metody polecanej dla małych systemów. Najczęściej używana jest aproksymacja Padégo [8], [9].

W powyższych rozważaniach opuściliśmy stałą czasową τ. Można ją łatwo wstawić do uzyskanego rozwiązania, ponieważ $V_m^T(A\tau)V_m = H_m$, a podprzestrzenie Krylowa związane z A i z $A\tau$ są identyczne.

Podsumowując, metoda podprzestrzeni Kryłowa dla stanów nieustalonych sprowadza się do:

- użycia procedury Arnoldiego do konstrukcji ortonormalnego zbioru wektorów bazowych V_m i górnej macierzy Hessenberga H_m,
- użycia aproksymacji Padégo do obliczenia $e^{H_m \tau}$,
- obliczenia poszukiwanego wektora prawdopodobieństw stanów, przybliżonego przez $\beta V_m e^{H_m \tau} e_1$.

Należy podkreślić, że jedynym działaniem związanym z macierzą A (czyli Q^T) jest mnożenie macierz×wektor w pierwszej linii drugiego kroku procedury Arnoldiego. Dlatego też metoda ta jest szczególnie polecana dla dużych, rzadkich systemów. Dokładność rozwiązania jest uzależniona od doboru kroku całkowania i rozmiaru m podprzestrzeni Kryłowa.

Metoda podprzestrzeni Kryłowa używana jest nie tylko do rozwiązywania powyższego problemu (układ równań (2) – gdzie macierz współczynników jest dużą, rzadką macierzą). Używa się jej również do rozwiązywania dużych, rzadkich układów równań liniowych (układ równań (3)), do numerycznego rozwiązywania równań parabolicznych, do rozwiązywania nieliniowych układów równań różniczkowych. Wiele szczegółów dotyczących wykorzystania tej metody do obliczania stanów nieustalonych w markowowskich modelach systemów komputerowych można znaleźć w [8].

4. Znajdowanie wektora własnego macierzy metodą podprzestrzeni Kryłowa

Jeżeli wygenerowano macierz tranzycji Q, to poszukiwany wektor prawdopodobieństw p, będący rozwiązaniem stanu ustalonego rozpatrywanego systemu, otrzymamy po rozwiązaniu układu równań liniowych (3).

Macierz Q jest z reguły macierzą dużą i rzadką, dlatego też do rozwiązania układu (3) wygodnie jest stosować metody iteracyjne, nie zmieniające jej zawartości i umożliwiające przez to pamiętanie jej w formie upakowanej. Jednak powszechnie używane metody iteracyjne – Jacobiego, Gaussa-Seidela czy nadrelaksacji nie gwarantują w przypadku macierzy Q zbieżności i w praktyce są często wolnozbieżne. Dlatego też przekształca się układ (3) do postaci:

$$W^T p = p, \quad \text{gdzie} \quad W^T = Q^T \Delta t + I \tag{8}$$

Macierz I jest macierzą jednostkową, a wartość $\Delta t \leqslant (max_i|q_{ii}|)^{-1}$ jest dobrana tak, by W^T była macierzą stochastyczną. Macierz stochastyczna to macierz kwadratowa, której elementy są nieujemne, suma elementów w każdym wierszu jest równa jedności, a suma elementów w każdej kolumnie jest dodatnia. Największa wartość bezwzględna wartości własnych macierzy stochastycznej jest równa jedności i poszukiwanie wektora p staje się poszukiwaniem lewego wektora własnego, odpowiadającego dominującej wartości własnej macierzy W^T.

Metoda podprzestrzeni Kryłowa polega na przekształceniu macierzy stochastycznej W^T o wymiarze n ($n \gg 1$) na macierz H_m, o znacznie mniejszym wymiarze m, której wektor własny będzie przybliżeniem wektora własnego macierzy W^T.

Macierz H, która jest górną macierzą Hessenberga o wymiarze $m \times m$ (m – rozmiar podprzestrzeni Kryłowa) oraz zbiór wektorów bazowych podprzestrzeni Kryłowa V_m wyznaczamy korzystając z procedury Arnoldiego, opisanej w poprzednim rozdziale.

Niech w jest wektorem własnym macierzy H_m. Możemy wtedy zapisać:

$$H_m w = w$$

Korzystając z równania (7) możemy powyższą zależność przekształcić do postaci:

$$\begin{aligned} V_m^T W^T V_m w &= w \\ W^T V_m w &= V_m w \end{aligned} \tag{9}$$

Z równań (8) i (9) wynika:

$$p = V_m w$$

Wektor własny w odpowiadający wartości własnej bliskiej jeden dla macierzy H_m znajdujemy metodą ilorazu Raleigha. Wektor ten po pomnożeniu przez macierz V_m daje nam wektor własny macierzy W^T, będący rozwiązaniem problemu.

5. Przykłady numeryczne

Algorytmy rozwiązywania stanów ustalonych w markowowskich modelach systemów komputerowych zostały zaimplementowane w pakiecie programowym AMOR [1]. Algorytmy dotyczące stanów nieustalonych nie są jeszcze dostępne w postaci gotowego pakietu, ale są już zaimplementowane, a skuteczność ich działania ilustruje poniższy przykład.

Rozpatrujemy parametry pracy sieci przedstawionej na rys. 5. W sieci tej krąży sześciu klientów. Liczba stanów n, jakie może przyjmować system z rys. 5, wynosi 16464. Jest to zarazem rozmiar macierzy Q ($n \times n$). Macierz ta jest więc bardzo duża, a zarazem rzadka, gdyż ilość elementów niezerowych wynosi 117495 (ilość wszystkich elementów wynosi 271063296).

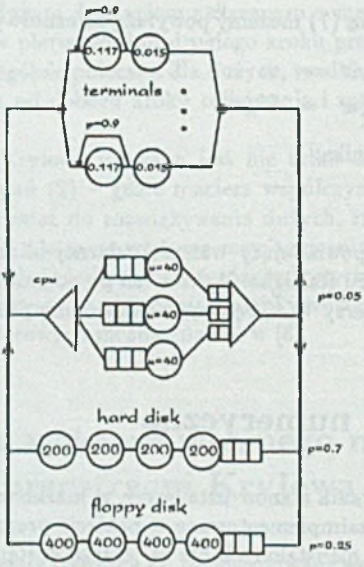
Na podstawie wektora prawdopodobieństw stanów (uzyskanego metodą podprzestrzeni Kryłowa) można obliczyć poszukiwane parametry systemu, np.:

- $E(n)$ – średnią liczbę klientów na stanowisku,
- T – przepustowość,
- $E(R)$ – średni czas odpowiedzi (czas czekania w kolejce + czas obsługi),
- U – wykorzystanie stanowiska.

Poniżej przedstawiono parametry systemu uzyskane dla:

- stanu ustalonego:

```
floppy_disk E(n)= 0.036948 T= 1.816174 E(R)= 0.020344  U= 0.036323
hard_disk   E(n)= 0.051996 T= 5.085289 E(R)= 0.010225  U= 0.050853
Main-CPU    E(n)= 0.385777 T= 7.264698 E(R)= 0.053103  U= 0.310969
cpu1        E(n)= 0.214926 T= 7.264698 E(R)= 0.029585  U= 0.181617
join-CPU1   E(n)= 0.170850 T= 7.264698 E(R)= 0.023518  U= 0.148117
cpu2        E(n)= 0.214926 T= 7.264698 E(R)= 0.029585  U= 0.181617
```



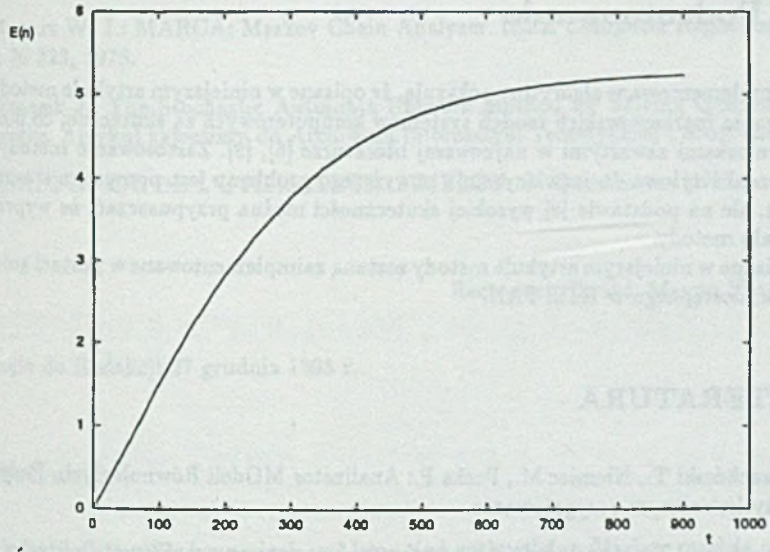
Rys. 5. Schemat przykładowego systemu komputerowego
Fig. 5. A sample computer system diagram

```
join-CPU2   E(n)= 0.170850 T= 7.264698 E(R)= 0.023518  U= 0.148117
cpu3        E(n)= 0.214926 T= 7.264698 E(R)= 0.029585  U= 0.181617
join-CPU3   E(n)= 0.170850 T= 7.264698 E(R)= 0.023518  U= 0.148117
terminals   E(n)= 5.525279 T= 0.363178 E(R)= 15.213676 U= 0.999996
```

- $t = 51$:

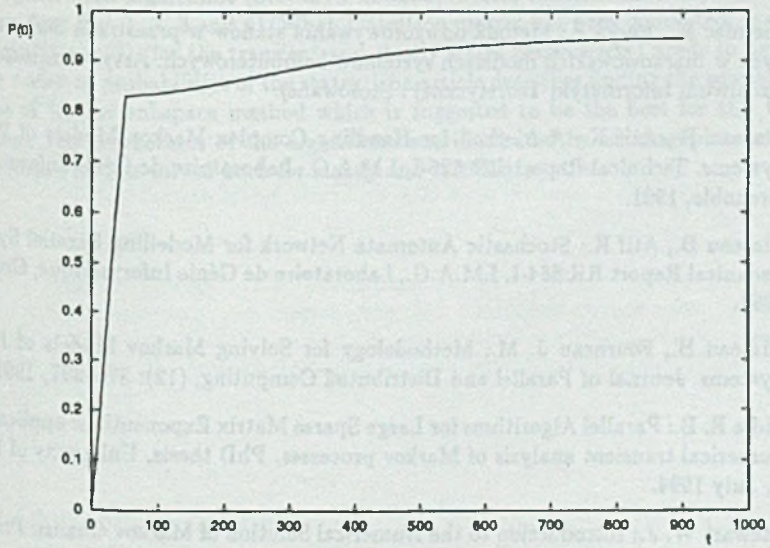
```
floppy_disk E(n)= 0.190845 T= 8.590137  E(R)= 0.022217  U= 0.171803
hard_disk   E(n)= 0.278990 T= 24.044872 E(R)= 0.011603  U= 0.240449
Main_CPU    E(n)= 4.783728 T= 33.798412 E(R)= 0.141537  U= 0.999475
cpu1        E(n)= 2.876123 T= 33.798412 E(R)= 0.085096  U= 0.844960
join-CPU1   E(n)= 1.907605 T= 33.798412 E(R)= 0.056441  U= 0.614512
cpu2        E(n)= 2.876123 T= 33.798412 E(R)= 0.085096  U= 0.844960
join-CPU2   E(n)= 1.907605 T= 33.798412 E(R)= 0.056441  U= 0.614512
cpu3        E(n)= 2.876123 T= 33.798412 E(R)= 0.085096  U= 0.844960
join-CPU3   E(n)= 1.907605 T= 33.798412 E(R)= 0.056441  U= 0.614512
terminals   E(n)= 0.746438 T= 0.049064  E(R)= 15.213676 U= 0.531742
```

Można też przedstawić uzyskane wyniki w postaci wykresów zmienności w czasie, np.: średnia liczba klientów na terminalach (patrz rys. 6) czy też prawdopodobieństwo, że stanowisko *floppy disk* jest wolne (patrz rys. 7).



Rys. 6. Średnia liczba klientów $E(n)$ na terminalach w poszczególnych chwilach czasowych

Fig. 6. Average number of clients $E(n)$ at *terminals* in particular moments of time



Rys. 7. Prawdopodobieństwo, że stanowisko *floppy disk* jest puste, w zależności od czasu

Fig. 7. Probability that *floppy disk* is vacant versus time

6. Podsumowanie

Zaimplementowane algorytmy pokazują, że opisane w niniejszym artykule metody rozwiązywania markowowskich modeli systemów komputerowych są skuteczne, co pokrywa się z wnioskami zawartymi w najnowszej literaturze [8], [9]. Zastosowanie metody podprzestrzeni Krylowa do rozwiązywania powyższego problemu jest pomysłem stosunkowo nowym, ale na podstawie jej wysokiej skuteczności można przypuszczać, że wyprze ona pozostałe metody.

Opisane w niniejszym artykule metody zostaną zaimplementowane w postaci gotowego pakietu, dostępnego w IITiS PAN.

LITERATURA

[1] Czachórski T., Niemiec M., Pecka P.: Analizator MOdeli Równoległych. Podręcznik użytkownika, w przygotowaniu.

[2] Czachórski T.: Modele kolejkowe systemów komputerowych. Skrypt Politechniki Śląskiej nr 1844, Gliwice 1994.

[3] Moler C. B., Van Loan C. F.: Nineteen dubious ways to compute the exponential of a matrix. SIAM Review., 20(4): 801-836, October 1978.

[4] Niemiec M., Pecka P.: Metoda odwzorowywania stanów w przestrzeń liczb naturalnych w markowowskich modelach systemów komputerowych. Artykuł zgłoszony do Archiwum Informatyki Teoretycznej i Stosowanej.

[5] Plateau B., Atif K.: A Method for Handling Complex Markov Models of Parallel Systems. Technical Report RR 826-I, I.M.A.G., Laboratoire de Génie Informatique, Grenoble, 1991.

[6] Plateau B., Atif K.: Stochastic Automata Network for Modelling Parallel Systems. Technical Report RR 854-I, I.M.A.G., Laboratoire de Génie Informatique, Grenoble, 1991.

[7] Plateau B., Fourneau J. M.: Methodology for Solving Markov Models of Parallel Systems. Journal of Parallel and Distributed Computing, (12): 370–387, 1991.

[8] Sidje R. B.: Parallel Algorithms for Large Sparse Matrix Exponentials: application to numerical transient analysis of Markov processes. PhD thesis, University of Rennes 1, July 1994.

[9] Stewart W. J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton, New Jersey 1994.

[10] Stewart W. J.: MARCA: Markov Chain Analyzer. IRISA Publication interne no. 45, Rennes, Juin 1976.

[11] Stewart W. J.: MARCA: Markov Chain Analyzer. IEEE Computer Repository, No. R 76 323, 1976.

[12] Tomasik J.: The Stochastic Automata Network Method for Solving Queueing Networks. Artykuł zgłoszony do Archiwum Informatyki Teoretycznej i Stosowanej.

[13] QNAP2 GUIDE DE L'UTILISATEUR. SIMULOG, St-Quentin sur Yvelines, 1992.

Recenzent: Dr inż. Marcin Skowronek

Wpłynęło do Redakcji 27 grudnia 1995 r.

Abstract

The article describes numerical problems tied with solving Markov models of computer networks. The solution may be reached either for the steady stade or at a specific moment of time. In both cases it is necessary to find the transition matrix containing the transition coefficients between particular states. The article describes various transition matrix generation algorithms (Stewart's method, inverse function method) and compares them (see Fig. 1, 2, 3 and 4). When transition matrix has been generated, system of equations (2) or (3) (for the transient and steady states respectively) needs to be solved for the vector of probabilities of the states. The article describes finding the solution with the use of Krylov subspace method which is suggested to be the best for this kind of problems. The correctness of the algorithms was illustrated by an example: a network model from Fig. 5 is solved both for steady and transient states.