STUDIA INFORMATICA 2014
Volume 35 Number 1 (115)

Marcin LAWNIK
Politechnika Śląska, Instytut Matematyki

GENEROWANIE PERMUTACJI Z WYKORZYSTANIEM ODWZOROWAŃ CHAOTYCZNYCH

Streszczenie. Prezentowany w tym artykule algorytm służy do generacji permutacji zbiorów skończonych, z wykorzystaniem odwzorowań chaotycznych. Jego zmodyfikowana wersja może być wykorzystana do generacji permutacji z zadanymi wcześniej wagami.

Słowa kluczowe: permutacje, permutacje z wagami, teoria chaosu

GENERATION OF PERMUTATIONS BASED UPON CHAOTIC MAPS

Summary. Presented in this article algorithm is used to generate permutations of finite set basing upon chaotic maps. The modified version of this algorithm may be used to generate a permutation with previously predetermined weights.

Keywords: permutation, permutation with weights, chaos theory

1. Wprowadzenie

Permutacje zbiorów skończonych znajdują sporo zastosowań w różnych dziedzinach nauki, takich jak kryptografia, symulacje czy sztuczna inteligencja. Często w zastosowaniach wymaga się dodatkowo, aby elementy permutacji miały przyporządkowane pewne wagi, które określałyby prawdopodobieństwo pojawienia się danego elementu. Taki typ permutacji jest szczególnie potrzebny w algorytmach związanych z metodami sztucznej inteligencji, jak algorytmy genetyczne czy algorytmy mrówkowe. Ponadto, wszelkiego rodzaju loterie czy gry komputerowe korzystają z algorytmów generowania permutacji z zadanymi wcześniej wagami.

Większość prezentowanych w literaturze algorytmów generowania permutacji nie pozwala na otrzymanie permutacji o różnych wagach przyporządkowanych poszczególnym elementom [1-9].

W tym artykule zaprezentowano nowy algorytm generowania permutacji oraz jego modyfikację, pozwalającą generować permutacje z zadanymi wagami poprzez wykorzystanie ciągu wartości odwzorowania chaotycznego.

1.1. Opis problemu

Niech M będzie zbiorem N-elementowym. Każdemu elementowi $m_i \in M$ przyporządkowana zostaje pewna liczba $P(m_i)$, zwana wagą, taka że $\sum_{i=1}^{N} P(m_i) = 1$. $P(m_i)$ oznacza prawdopodobieństwo wylosowania elementu m_i. W przypadku gdy $P(m_i) = \frac{1}{N}$ dla każdego $i = 1,2,\ldots,N$, elementy zbioru mają takie samo prawdopodobieństwo pojawienia się na danej pozycji w permutacji. Gdy wagi $P(m_i)$ nie są sobie równe, to mamy przypadek permutacji z zadanymi wagami, co oznacza, że niektóre elementy mają większą szansę pojawienia się na początkowych pozycjach niż pozostałe.

1.2. Odwzorowanie chaotyczne

Podstawowym wymaganiem prezentowanych algorytmów jest wykorzystanie odwzorowania chaotycznego o jednostajnym rozkładzie wartości zmiennej iterowanej. Wymaganie to spełnia m.in. *asymetryczne odwzorowanie namiotowe*, dane wzorem:

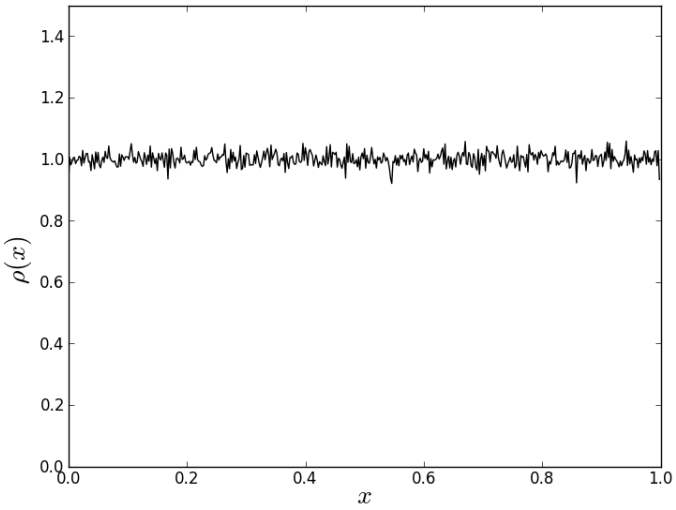
$$f_p(x) = \begin{cases} x/p, & x \in [0, p] \\ (1-x)/(1-p), & x \in (p,1] \end{cases} \tag{1}$$

gdzie $x \in [0,1]$ i $p \in (0,1)$. Dla każdej wartości $p \in (0,1)$ powyższe odwzorowanie jest chaotyczne. Jego rozkład uzyskany numerycznie pokazany jest na rys. 1.

2. Algorytmy

2.1. Generowanie permutacji

1) Stwórz tablicę przyporządkowań elementów zbioru do atraktora odwzorowania chaotycznego, dzieląc go na N podprzedziałów o równej długości.



Rys. 1. Rozkład asymetrycznego odwzorowania namiotowego
Fig. 1. Skew tent map distribution

2) Korzystając z danego odwzorowania chaotycznego, jego parametrów i warunku początkowego wygeneruj wartość odwzorowania, która, wpadając w odpowiedni podprzedział, wskazuje znak związany z tym podprzedziałem. Znak ten jest aktualnym elementem permutacji, a aktualna wartość odwzorowania staje się nowym warunkiem początkowym.

3) Zmodyfikuj tablicę przyporządkowań znaków do podzbioru przestrzeni stanów, pomijając znak osiągnięty w **2)**, zwiększając tym samym długość każdego podprzedziału, tak aby nowa tablica pokrywała w całości rozmiar atraktora.

Obliczanie nowych wag polega na ustawieniu długości każdego podprzedziału na $\frac{1}{i-1}$, gdzie poprzednio mieliśmy dokładnie i elementów przyporządkowanych do podzbioru przestrzeni stanów.

4) Powtórz **2)** i **3)** N razy, otrzymując jako wynik permutację zbioru N-elementowego.

2.2. Generowanie permutacji z zadanymi wagami

1) Zadaj dla zbioru N-elementowego wagi dla poszczególnych jego elementów. Stwórz tablicę przyporządkowań znaków do podzbioru przestrzeni stanów odwzorowania chaotycznego, dzieląc go na N podprzedziałów o długości, która odpowiada proporcjonalnie wadze danego elementu.

2) Korzystając z danego odwzorowania chaotycznego, jego parametrów i warunku początkowego wygeneruj wartość odwzorowania, która, wpadając w odpowiedni podprzedział, wskazuje znak z nim związany. Znak ten jest aktualnym elementem permutacji, a aktualna wartość odwzorowania staje się nowym warunkiem początkowym.

3) Zmodyfikuj tablicę przyporządkowań elementów zbioru do atraktora, pomijając element osiągnięty w **2)**, zwiększając tym samym długość każdego podprzedziału, tak aby nowa tablica zachowywała zadane pozostałe proporcje wag i pokrywała cały podzbiór przestrzeni stanów.

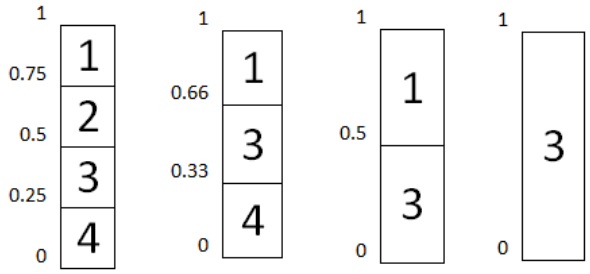
Obliczanie nowych wag polega na podzieleniu każdej pozostałej wagi przez liczbę $1-P(m_i)$, gdzie $P(m_i)$ jest wagą wybranego w *i*-tym kroku elementu.

4) Powtórz procedury z **2)** i **3)** N - razy, otrzymując jako wynik permutację zbioru *N*-elementowego z zadanymi wcześniej wagami.

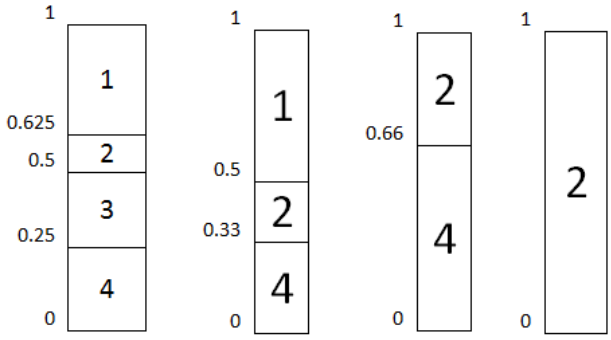
2.3. Przykłady

Przykład działania algorytmu generowania permutacji bez wag dla zbioru $\{1,2,3,4\}$ przedstawiono na rys. 2. Uzyskaną permutacją jest $(2,4,1,3)$. Oznacza to, że pierwszym wybranym elementem jest 2. W kolejnym kroku dla pozostałych elementów zbioru należy zmienić długość podprzedziału na $\frac{1}{3}$. Powyższe procedury kontynuujemy, aż wyjściowy zbiór będzie pusty.

Przykład działania algorytmu 2.2 generowania permutacji z wagami dla zbioru $\{1,2,3,4\}$ z zadanymi wagami $\left\{\frac{3}{8},\frac{1}{8},\frac{2}{8},\frac{2}{8}\right\}$ przedstawiono na rys. 3. Uzyskaną permutacją jest $(3,1,4,2)$. Po wyborze pierwszego elementu należy odpowiednio przeskalować pozostałe wagi, tak aby zachowywały poprzednie zależności i nadal sumowały się do jedynki. Należy zatem podzielić każdą z pozostałych wag przez $1-\frac{2}{8}$. Nowymi wartościami wag dla pozostałego zbioru są odpowiednio $\left\{\frac{3}{6},\frac{1}{6},\frac{2}{6}\right\}$.



Rys. 2. Przykład działania algorytmu 2.1
Fig. 2. Example of generating permutations with algorithm 2.1



Rys. 3. Przykład działania algorytmu 2.2
Fig. 3. Example of generating permutations with algorithm 2.2

3. Analiza

Analizy prezentowanej metody generacji permutacji dokonano poprzez zaproponowane w literaturze kryteria:

1) sprawdzenie, czy otrzymana permutacja ma element stały, tzn. $q(i) = i$, gdzie $q(i)$ jest *i*-tym elementem permutacji;
2) sprawdzenie, czy sąsiadujące elementy są zachowane przy tworzeniu permutacji, tzn. czy $q(i) + 1 = q(i+1)$ dla $i = 1,2,\ldots,N-1$ [10];
3) wyliczenie współczynnika przesunięć [11,12] określonego za pomocą wzoru:

$$\alpha = \frac{1}{N}\sum_{i=1}^{N}\left|i - q(i)\right|. \tag{2}$$

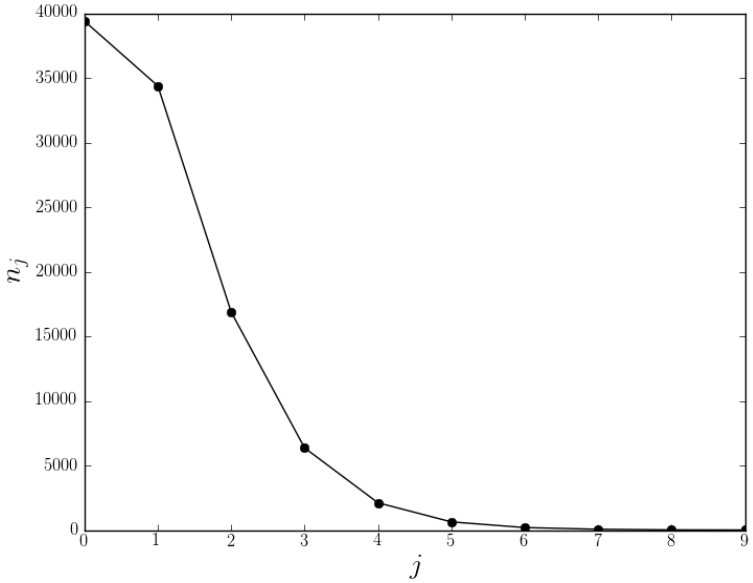
W zastosowaniach liczba permutacji, które pozytywnie spełniają 1), 2) oraz $\alpha < N/3$, powinna być jak najmniejsza. Wyniki dla 100000 wygenerowanych permutacji za pomocą algorytmu 2.1 o długości 256 zamieszczono w tabeli 1.

Tabela 1

Procent permutacji, które nie spełniają warunków 1 i 2 oraz ich współczynnik przesunięcia jest większy niż $N/3$

1	2	3
37.692%	66.939%	58.037%

Przeprowadzono również analizę wrażliwości wygenerowanych permutacji na zmianę warunku początkowego za pomocą algorytmu 2.1. Dla 100000 permutacji, zmieniając warunek początkowy o $\delta = 10^{-15}$, sprawdzono, ile elementów zachowało swoją dotychczasową pozycję. Wyniki pokazano na rys. 4.



Rys. 4. Liczba permutacji n_j, które zachowały j elementów na tej samej pozycji przy zmianie warunku początkowego o $\delta = 10^{-15}$

Fig. 4. Number of permutations n_j, which preserve j elements in the same position with the change of the initial condition by $\delta = 10^{-15}$

4. Wnioski

Pokazany w artykule algorytm pozwala na generowanie permutacji zbiorów N-elementowych. Jego zmodyfikowana wersja może służyć do otrzymywania permutacji zbiorów N-elementowych o zadanych wcześniej wagach. Obie procedury wykorzystują odwzorowanie chaotyczne o rozkładzie jednostajnym.

BIBLIOGRAFIA

1. Knuth D.: The Art of Computer Programming volume 2: Seminumerical algorithms. Addison–Wesley, 1969.
2. Lipski W.: Kombinatoryka dla programistów. WNT, Warszawa 1982.
3. Sattolo, S.: An algorithm to generate a random cyclic permutation. Inf. Process. Lett. 22, 1986.
4. Arndt J.: Generating Random Permutations, rozprawa doktorska
5. Kuo T.: A new method for generating permutations in lexicographic order. Journal of Science and Engineering Technology, Vol. 5, No. 4, 2009.

6. Johnson S.: Generation of permutations by adjacent transpositions. Mathematics of Computation, No. 17 (83), 1963.
7. Martinez C., Panholzer A., Prodinger H.: Generating random derangements. Proceedings of the 10th ACM-SIAM Workshop on Algorithm Engineering and Experiments (ALENEX) and the 5th ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics (ANALCO), 2008.
8. Dăscălescu A., Boriga R.: A novel fast chaos-based algorithm for generating random permutations with high shift factor suitable for image scrambling. Nonlinear Dynamics, 2013.
9. Awad A., Saadane A.: New Chaotic Permutation Methods for Image Encryption. IAENG Int. J. Comput. Sci. 37(4), 2010.
10. Ravichandran V., Srinivasan N., Jayamala M., Sivagurunathan S.: Permutation for speech scrambling. J. Indian Acad. Math. 25(1), 2003.
11. Mitra A., Subba Rao, Y., Prasanna, S.: A new image encryption approach using cmbinational permutation techniques. Int. J. Comput. Sci. 1(2), 2006.
12. Becker H., Piper F.: Secure Speech Communications. Academic Press, London 1985.

Abstract

This article presents an algorithm for generation of permutations. Modification of this algorithm can be used to generate permutations with given weights. Both procedures are based on chaotic maps with uniform distribution such as (1). The first step in each algorithm is to assign the elements of the set to the attractor. Iterating then the chaotic map we get an value, which falls into one of the sub-intervals. Thus the element of the set, which is assigned to this sub-interval, is the current permutation element. In next step after removing the current permutation element from the attractor, a new assignment of the left elements to the attractor must be made. In the case of permutations with given weights, a new assignment must also behave previous weight proportions of the left elements. Above procedure must be continue until a permutation is generated. Examples of generation permutations with presented algorithms are shown in fig. 1 and fig. 2.

Adres

Marcin LAWNIK: Politechnika Śląska, Instytut Matematyki, ul. Kaszubska 23,
44-100 Gliwice, Polska, marcin.lawnik@polsl.pl