

# ANALIZA ALGORYTMÓW, LAB 3

Adrian Mucha, Politechnika Wrocławska, WPPT

25/04/2020

## Zad 9

Generowane multizbiory (każdy eksperyment, dla każdego  $n$ ) zawierają zdarzenia niezależne od siebie. Każdy element posiada unikalne id, które są kolejnymi liczbami naturalnymi.

Wartości cech są generowane według zdefiniowanych parametryzowanych strategii, które zostały podzielone na sekcje i opisane poniżej.

Do uzyskania losowości funkcji haszującej rzutującej na zbiór  $[0, 1]$  użyto funkcji md5, sha1, sha3. Zbadano wyniki dla różnych wartości parametru  $m$  oznaczającego ilość rejestrów.

Zgodnie z intuicją, obserwujemy że im większy parametr  $m$ , tym dokładniejsze są wartości aproksymacji sumy.

Dobre funkcje haszujące nie dają różnicy pomiędzy eksperymentami. Należy jednak zaznaczyć, że ich odporność na kolizje ma znaczący wpływ na ogólną precyzję algorytmu.

## Nierówność Czebyszewa

$$P(|X - \mathbb{E}(X)| < \delta) > 1 - \alpha$$

$$X = \frac{\hat{\Lambda}}{\Lambda}, \mathbb{E}(X) = 1$$

$$\alpha = \frac{\text{Var}(\frac{\hat{\Lambda}}{\Lambda})}{\delta^2}, \text{Var}(\frac{\hat{\Lambda}}{\Lambda}) = \frac{1}{m-2}$$

$$\alpha = \frac{1}{m-2} \delta^2, \delta = \sqrt{\frac{1}{\alpha(m-2)}}$$

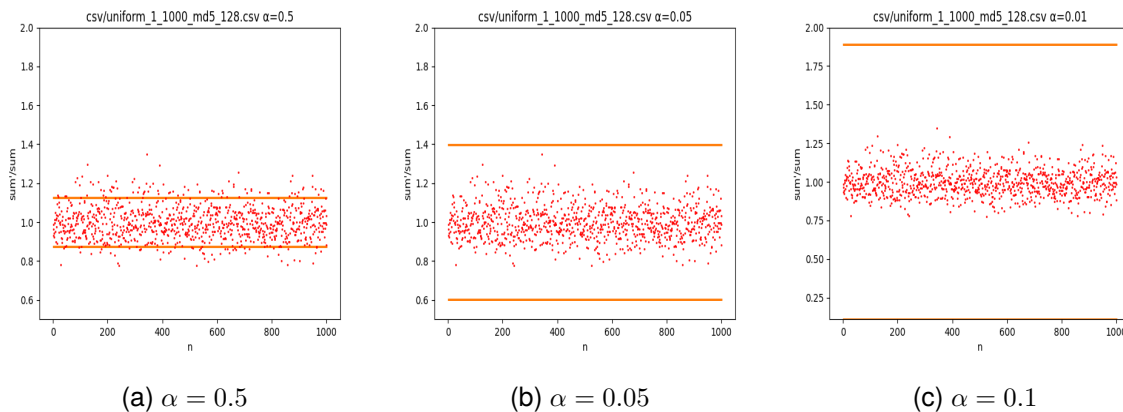
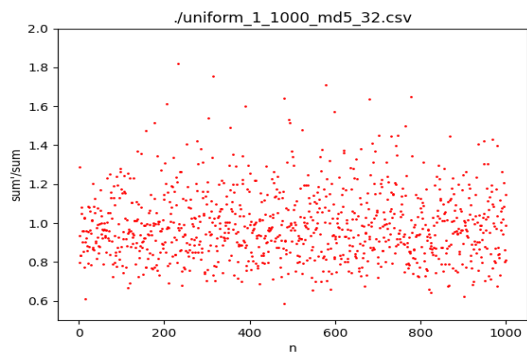


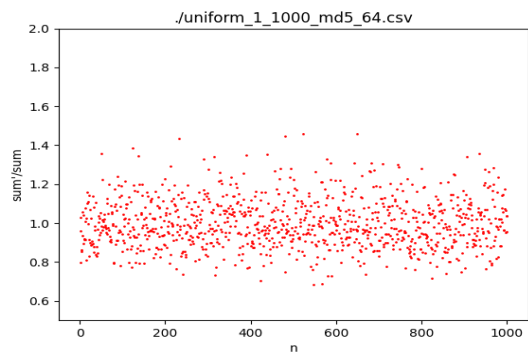
Figure 1: Wykresy przedstawiają nierówności Czebyszewa nałożone na wykresy z różnymi parametrami  $\alpha$ . Możemy zaobserwować, że ograniczenia uzyskane z nierówności prawidłowo odzwierciedlają otrzymane wyniki.

## Rozkład jednostajny na $[1, 1000]$

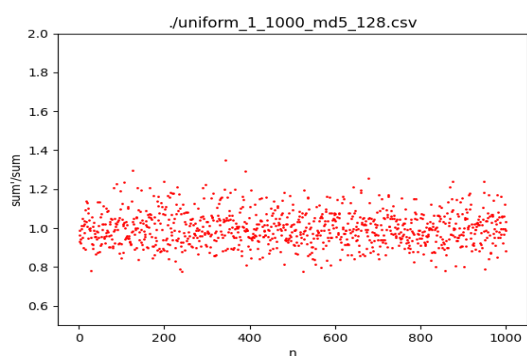
Strategia generowania cech, to próbkowanie z zakresu  $[1, 1000]$ .



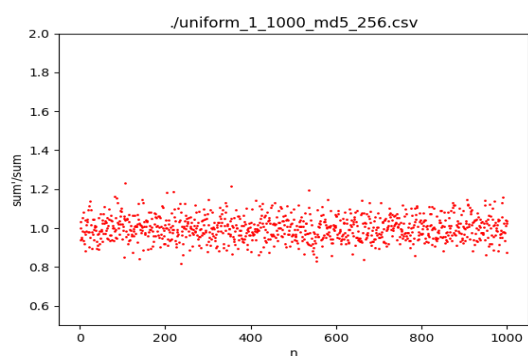
(a)  $m = 32$



(b)  $m = 64$

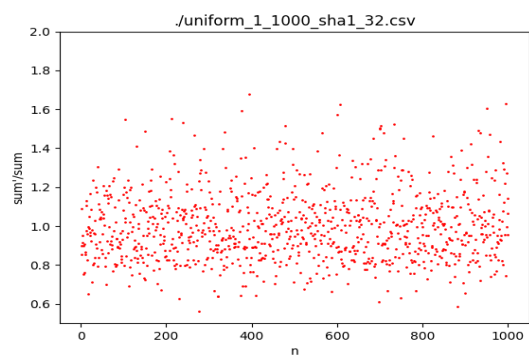


(c)  $m = 128$

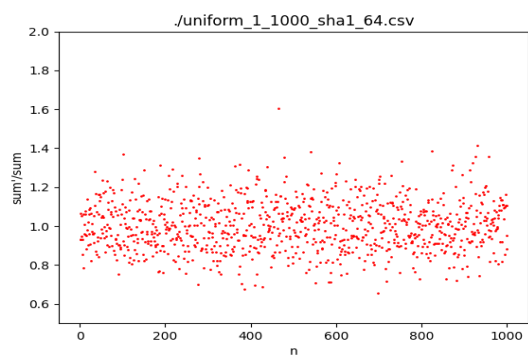


(d)  $m = 256$

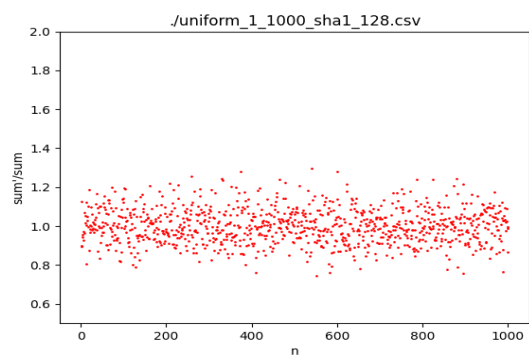
Figure 2: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech pochodzą z rozkładu jednostajnego na przedziale  $[1, 1000]$ . Wykorzystany algorytm: md5.



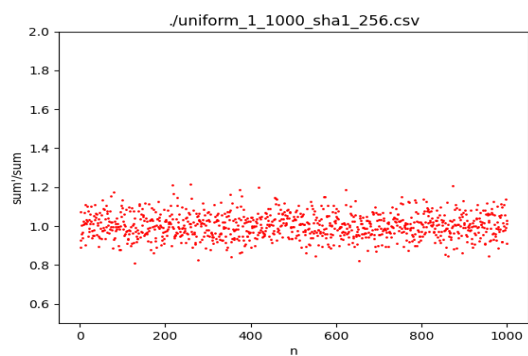
(a)  $m = 32$



(b)  $m = 64$

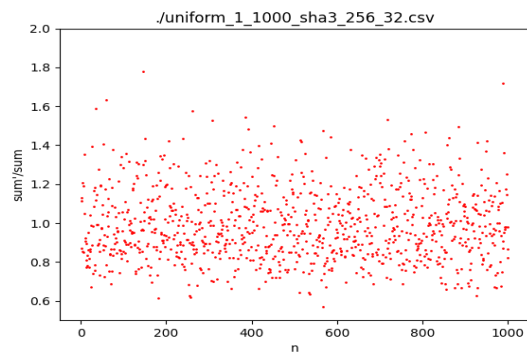


(c)  $m = 128$

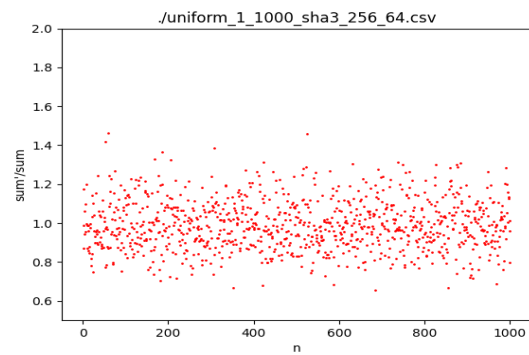


(d)  $m = 256$

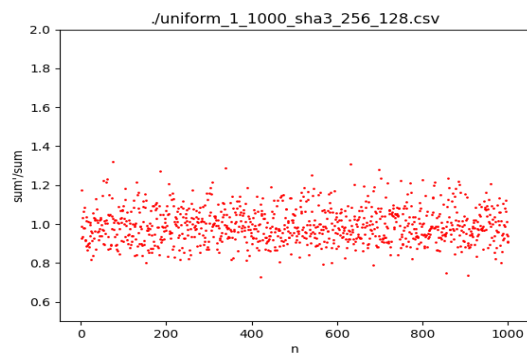
Figure 3: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech pochodzą z rozkładu jednostajnego na przedziale  $[1, 1000]$ . Wykorzystany algorytm: sha1.



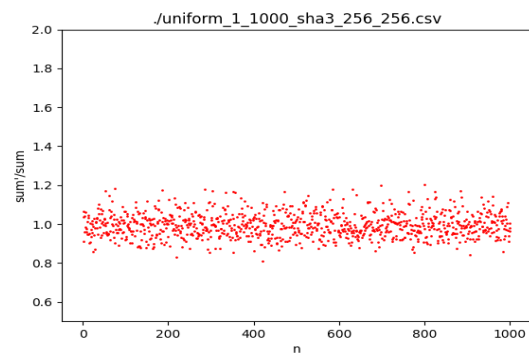
(a)  $m = 32$



(b)  $m = 64$



(c)  $m = 128$

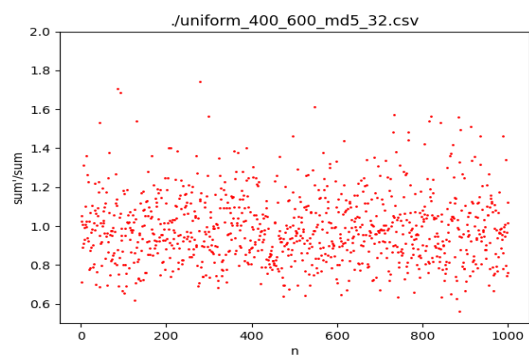


(d)  $m = 256$

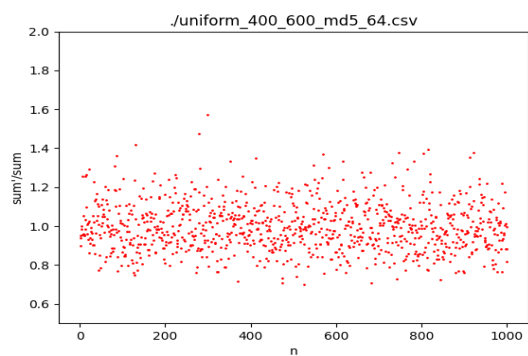
Figure 4: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech pochodzą z rozkładu jednostajnego na przedziale  $[1, 1000]$ . Wykorzystany algorytm: sha3.

### Rozkład jednostajny na $[400, 600]$

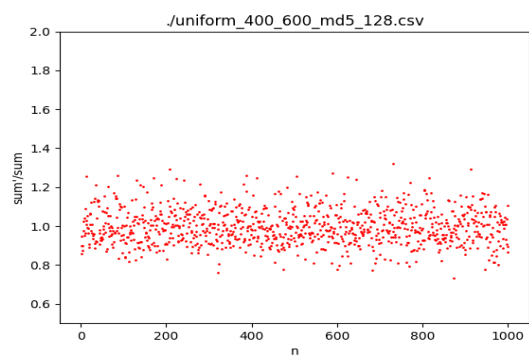
Strategia generowania cech, to próbkowanie z zakresu  $[400, 600]$ . Brak znaczących różnic w porównaniu do zbioru  $[1, 1000]$ .



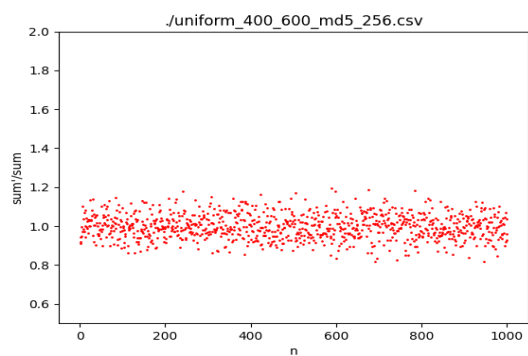
(a)  $m = 32$



(b)  $m = 64$

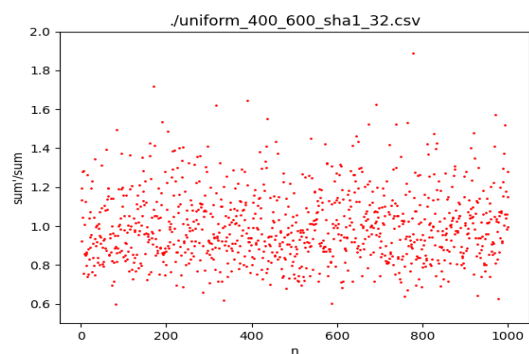


(c)  $m = 128$

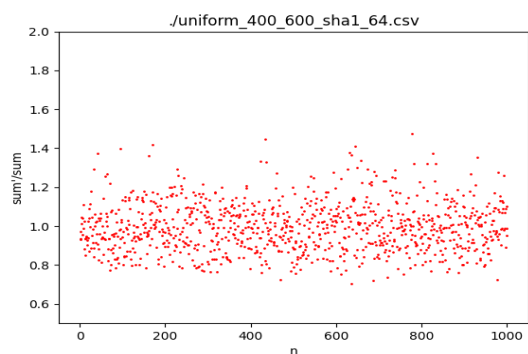


(d)  $m = 256$

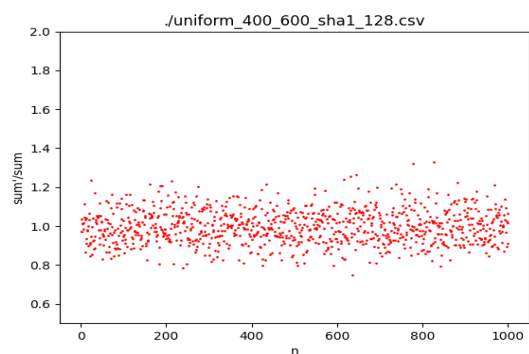
Figure 5: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech pochodzą z rozkładu jednostajnego na przedziale  $[400, 600]$ . Wykorzystany algorytm: md5.



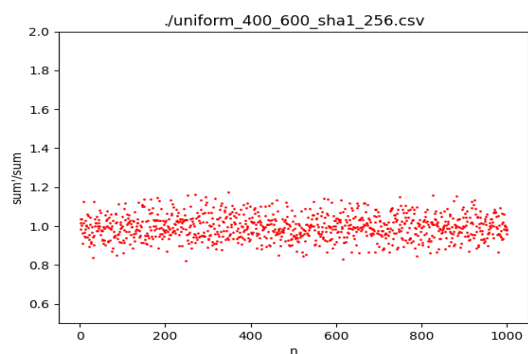
(a)  $m = 32$



(b)  $m = 64$

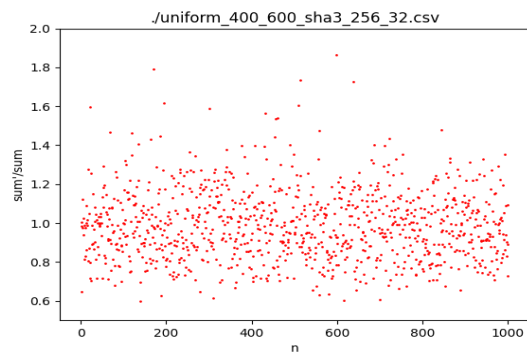


(c)  $m = 128$

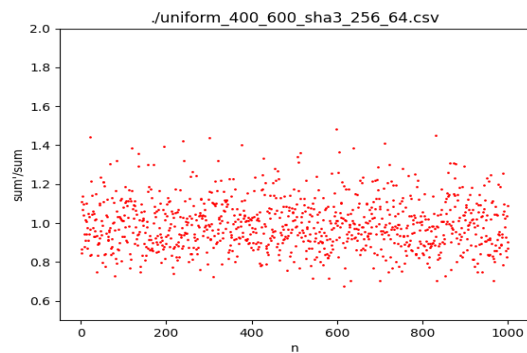


(d)  $m = 256$

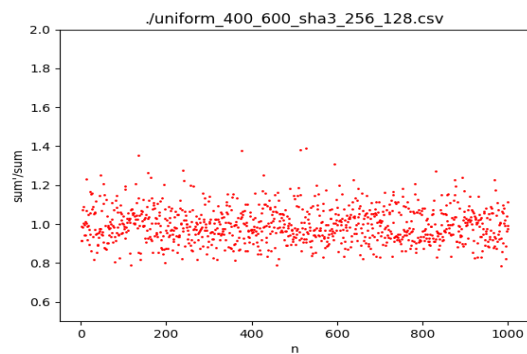
Figure 6: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech pochodzą z rozkładu jednostajnego na przedziale  $[400, 600]$ . Wykorzystany algorytm: sha1.



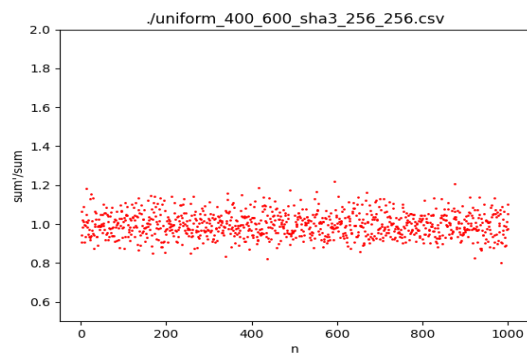
(a)  $m = 32$



(b)  $m = 64$



(c)  $m = 128$

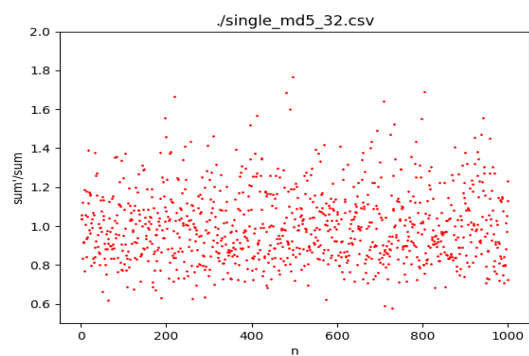


(d)  $m = 256$

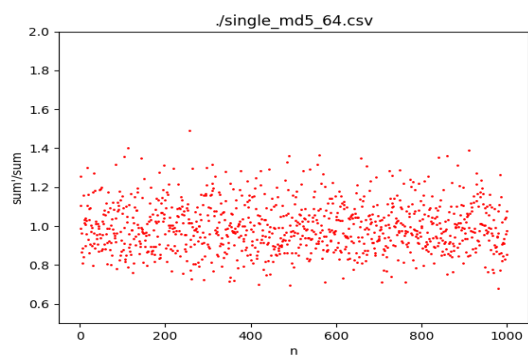
Figure 7: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech pochodzą z rozkładu jednostajnego na przedziale  $[400, 600]$ . Wykorzystany algorytm: sha3.

### Jednakowe cechy

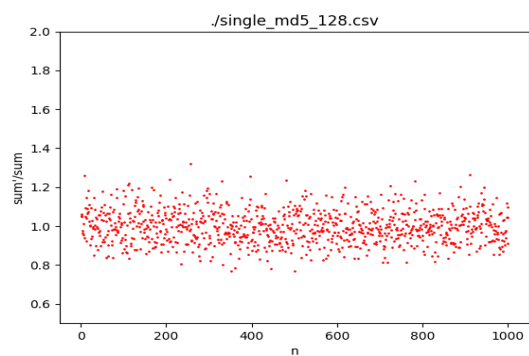
W tej strategii wszystkie cechy są ustawione na 1. W tym przypadku algorytm zachowuje się podobnie jak MinCount i zlicza unikalne wystąpienia, lecz daje gorsze wyniki.



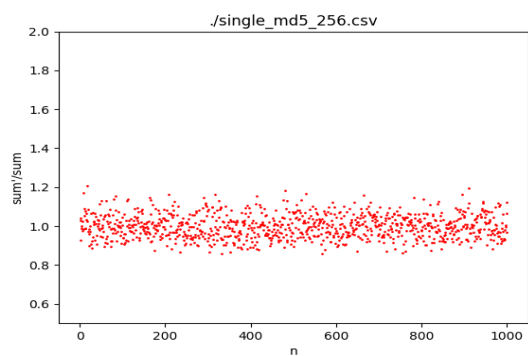
(a)  $m = 32$



(b)  $m = 64$



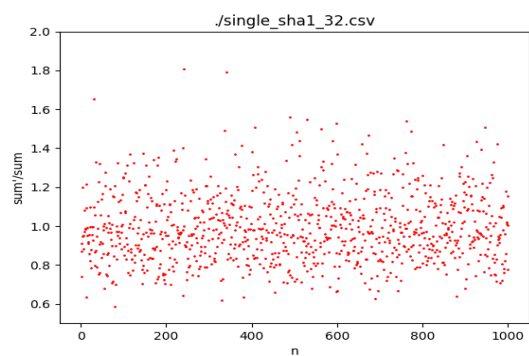
(c)  $m = 128$



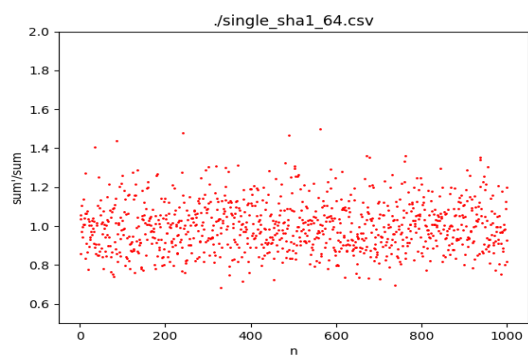
(d)  $m = 256$

Figure 8: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech są jednakowe (wynoszą 1). Wykorzystany algorytm: md5.

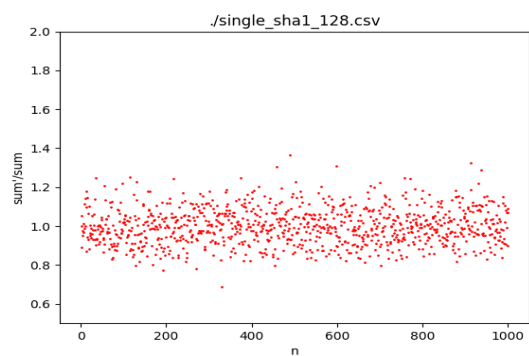




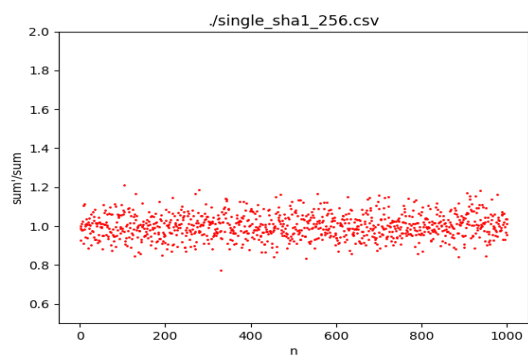
(a)  $m = 32$



(b)  $m = 64$

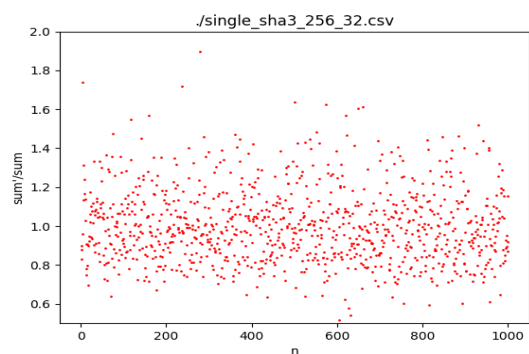


(c)  $m = 128$

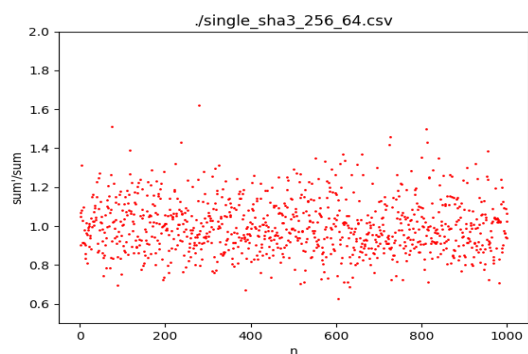


(d)  $m = 256$

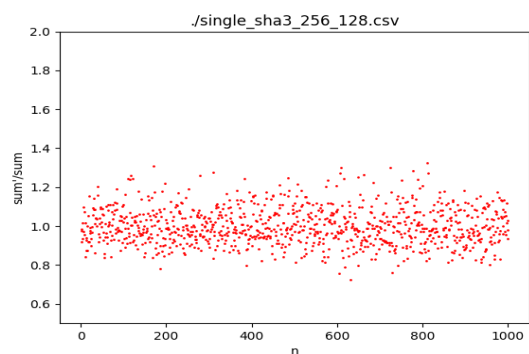
Figure 9: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech są jednakowe (wynoszą 1). Wykorzystany algorytm: sha1.



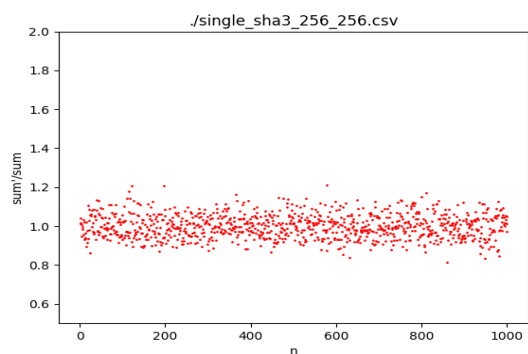
(a)  $m = 32$



(b)  $m = 64$



(c)  $m = 128$

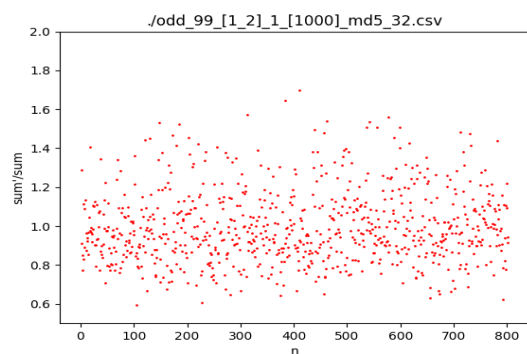


(d)  $m = 256$

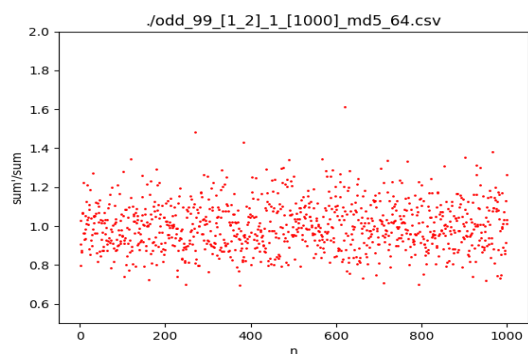
Figure 10: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech są jednakowe (wynoszą 1). Wykorzystany algorytm: sha3.

## Wartości odstające

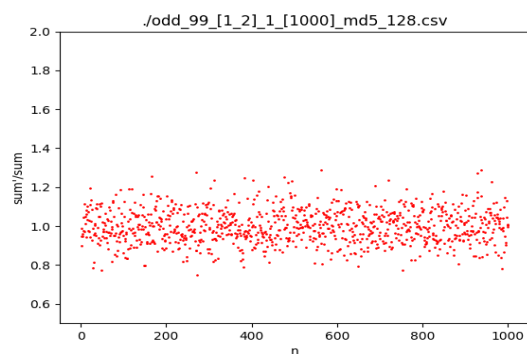
Wartość cechy losowana jest ze zbioru  $[1, 2, 1, 2, \dots]$ , którego 1% stanowi wartość odstająca (duża liczba, np. 1000).



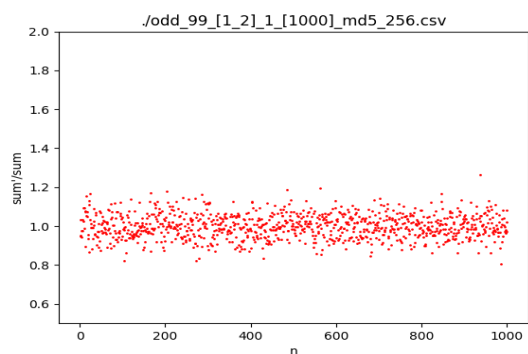
(a)  $m = 32$



(b)  $m = 64$

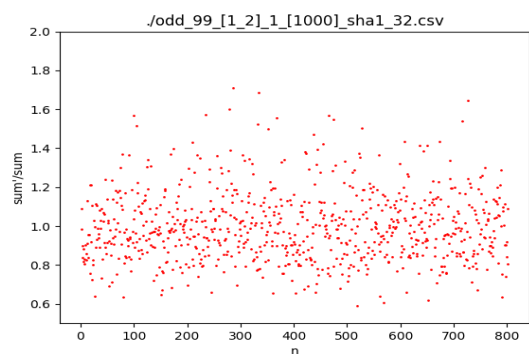


(c)  $m = 128$

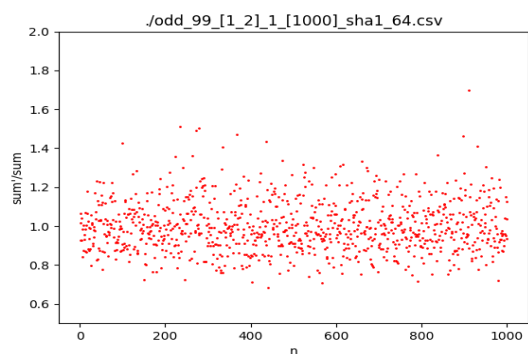


(d)  $m = 256$

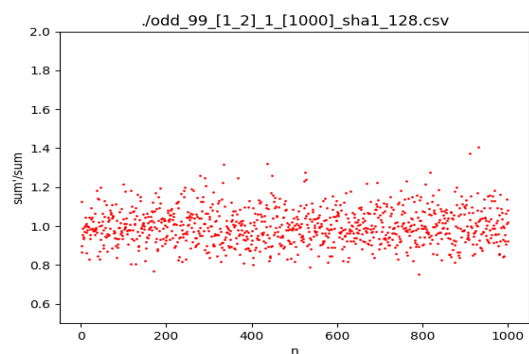
Figure 11: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech są w większości podobne, lecz zawierają wartości odstające. Wykorzystano stosunek, 1%. Wykorzystany algorytm: md5.



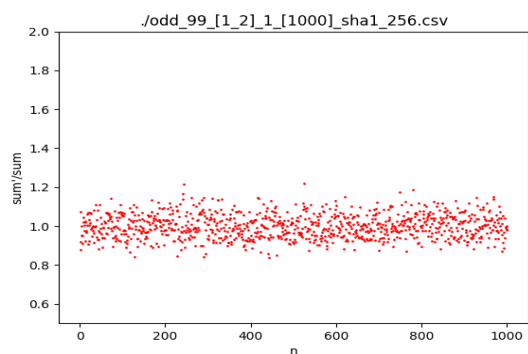
(a)  $m = 32$



(b)  $m = 64$

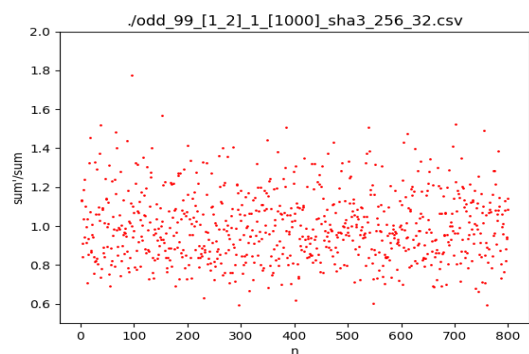


(c)  $m = 128$

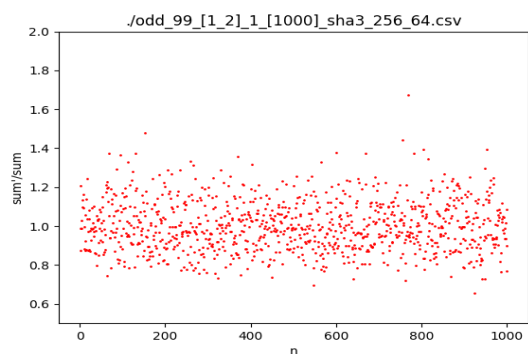


(d)  $m = 256$

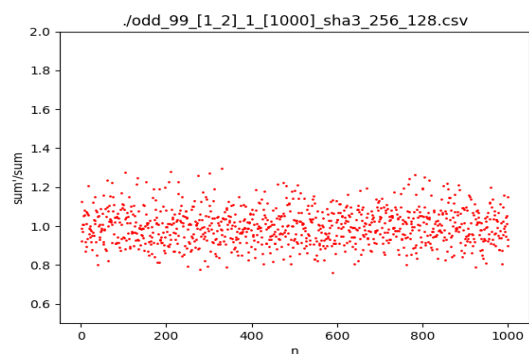
Figure 12: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech są w większości podobne, lecz zawierają wartości odstające. Wykorzystano stosunek, 1%. Wykorzystany algorytm: sha1.



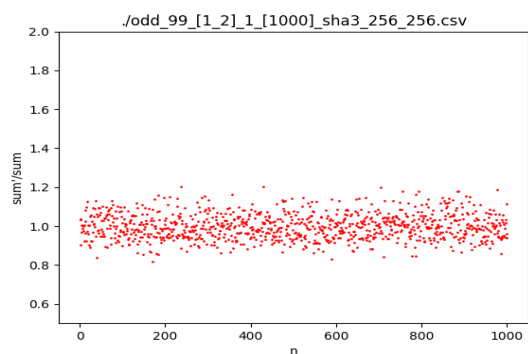
(a)  $m = 32$



(b)  $m = 64$



(c)  $m = 128$



(d)  $m = 256$

Figure 13: Wykresy przedstawiają wyniki algorytmu dla różnej ilości rejestrów gdy wartości cech są w większości podobne, lecz zawierają wartości odstające. Wykorzystano stosunek, 1%. Wykorzystany algorytm: sha3.