*Relazione al progetto*

I Parte

**Test di Uniformità**

Il gruppo ha deciso di sottoporre ad un Test di Uniformità (basato sulla statistica *Chi-Quadro)* il generatore di *Lehmer* con moltiplicatore e modulo rispettivamente *(a, m) = (48271, 2³¹ – 1).*

**Note Teoriche sul test**

Il test si basa sulla suddivisione dell’intervallo (0,1) (sul quale opera il generatore di *Lehmer*) in *k bins ,* sulla generazione di *n* chiamate a *random()* e il conseguente conteggio di osservazioni che cadono all'interno del *bin x-esimo* (∀*x*∈{0,1,…,*k*-1})*.*

Indichiamo con:

*o[x] := “*il numero di osservazioni che cadono nel *bin x-esimo”*;

*e[x] := “*il numero atteso di osservazioni che cadono nel *bin x-esimo* quando esse sono realizzazioni di variabili aleatorie *U*(0,1) (ossia *e[x]= n/k*)”.

Ebbene, quando il campione è un insieme fatto di realizzazioni di variabili aleatorie indipendenti e *U(0,1)* e quando *n/k* è sufficientemente grande (*n/k* ≥ 10) si dimostra che la seguente statistica tende ad una variabile aleatoria *Chi-Quadro(k-1)*:

Intuitivamente, quando *random()* genera effettivamente realizzazioni di variabili aleatorie *U(0,1)* la quantità al numeratore tende ad essere piccola (poiché *o[x]* tende ad *n/k*). Sotto questa ipotesi tutta la statistica vtende ad essere piccola. Per questo motivo, fissato un livello di confidenza *(1-α)* con *α* solitamente pari a *0.05*, e posti v1\* e v2\*:

v1\* = idfChisquare(*k*-1, *α*/2 ) e v2\* = idfChisquare(*k*-1, 1 - *α*/2 ), ovvero i quantili della *Chi-Quadro(k-1)* rispettivamente di ordine *α*/2 e 1 - *α*/2

Allora:

Pr(v1\* ≤ v ≤ v2\*) = 1- *α*

Per questo motivo si asserisce che se la statistica v è tale per cui v > v2\* o v < v1\* allora il test fallisce: questo accade evidentemente con probabilità *α* (di solito *0.05*) se la densità è effettivamente *U(0,1)*.

In effetti, come verrà illustrato nel seguito della trattazione, per testare l’uniformità del generatore sotto esame verranno prodotte *s* statistiche, una per ogni *stream* messo a disposizione dal generatore di *Lehmer* della libreria *rngs* (in tutto 256 *streams*). In caso di uniformità ci si aspetta il fallimento del test da parte di *s\*α* (256\*0.05 ≈ 13) statistiche: qualora ci si discostasse troppo da questo valore (per eccesso o per difetto) si solleverebbero profondi dubbi sull’uniformità del generatore.

**Motivazioni di impiego del test e relativi punti di forza e debolezza**

Certamente il primo test da sottoporre ad un generatore è il test di uniformità. Esso svolge un ruolo essenziale per l’identificazione di tutte le implementazioni non corrette già dal punto di vista della *flatness*. Tuttavia l’aver superato il test in questione da parte di un generatore non implica la sua bontà: ad esempio questo test non tiene affatto conto dell’ordine con il quale sono riempiti i *bins*. Per questo motivo viene detto test di randomicità statisticamente *debole*.

**Codice**

L’implementazione del test è stata sviluppata in linguaggio *C*, utilizzando le librerie *rngs* (generatore di *Lehmer multistream*)e *rvms* (quantili *Chi-Quadro*). Di seguito il codice nel file *chisquare.c*:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "rngs.h"

#include "rvms.h"

#define K 1000 /\*It is the number of bin (K >= 1000)\*/

#define N 10000 /\*It is the number of variate for STREAM (N >= 10\*K)\*/

#define ALFA 0.05 /\*Level of confidence of Chi-Square test\*/

#define STREAMS 256 /\*Considered STREAMS\*/

double chiSquareByStream(int index){

/\*Return the Chi-Square statistic based on N variates selected by STREAM index\*/

double expected = (N/K);

double statistic = 0;

int o[K];

int x;

int i;

for(x = 0 ; x < K ; x++)

o[x] = 0;

SelectStream(index);

for(i = 0 ; i < N ; i++){

x = floor((Random())\*K);

o[x]++;

}

for(x = 0 ; x < K ; x++){

statistic = statistic + (double)(pow((o[x] - expected) , 2)) / (expected);

}

return statistic;

}

double\* chiSquare(){

/\*Return all 256 Chi-Square statistics, one for STREAM\*/

double\* V = (double\*)malloc(STREAMS\*sizeof(double));

if(V==NULL) {

perror("Error in malloc\n");

exit(1);

}

int i;

for(i = 0 ; i < STREAMS ; i++)

V[i] = 0;

for(i = 0 ; i < STREAMS ; i++)

V[i] = chiSquareByStream(i);

return V;

}

double computeV1(void){

/\*Return idfChisquare(K âˆ’ 1, ALFA / 2)\*/

return idfChisquare((long)(K - 1), (double)(ALFA / 2));

}

double computeV2(void){

/\*Return idfChisquare(K âˆ’ 1, 1 - ALFA / 2)\*/

return idfChisquare((long)(K - 1), (double)(1 - (ALFA / 2)));

}

void destroy(double\* v){

/\*Free heap\*/

free(v);

}

int writeData(void){

/\*Write results of chiSquare() on data.txt"\*/

FILE\* fd;

int i;

fd = fopen("data.txt", "a");

if(fd==NULL) {

perror("Error opening file");

exit(1);

}

double\* V = chiSquare();

for(i = 0 ; i < STREAMS ; i++)

fprintf(fd, "%0.2f,", V[i]);

fprintf(fd, "%0.2f,", computeV1());

fprintf(fd, "%0.2f", computeV2());

destroy(V);

fclose(fd);

return 0;

}

int main(void)

{

int i = writeData();

if(i != 0)

return EXIT\_FAILURE;

else

return EXIT\_SUCCESS;

}

Il numero di *bin* è impostato a 1000 e dunque per rendere la statistica utilizzabile *n* è impostato a 10.000 (*n/k* ≥ 10). L’algoritmo costruisce un array di 256 statistiche (una per *stream*), salvate successivamente nel file *data.txt* nel formato *stat\_0, stat\_1, … , stat\_255* appendendo infine i due quantili v1\* e v2\*.

**Visualizzazione dello *scatter plot* e conclusioni**

I dati scritti sul file *data.txt* sono utilizzati dall’applicazione statistica *R* per visualizzare su uno *scatter plot* tutte le 256 statistiche ottenute e verificare il superamento o meno del test da parte del generatore di *Lehmer (a, m) = (48271, 2³¹ – 1)*. Nella pagina successiva è inserita l’immagine dello *scatter plot* ottenuto. Sulle ascisse vi è il numero dello *stream* considerato e sulle ordinate il valore della statistica ottenuta dal test di uniformità sul generatore associato.Le linee orizzontali che corrispondono rispettivamente ai punti sulle ordinate di valore circa 913.3 e 1088.5 rappresentano i quantili della *Chi-Quadro* v1\* e v2\*. È evidente che vi sono 8 punti minori del quantile v1\* (troppa uniformità) e 6 punti maggiori del quantile v2\* (assenza di uniformità). Dunque il test di uniformità in tutto fallisce 14 volte: tuttavia era stato precedentemente asserito che in caso di uniformità da parte del generatore ci si aspettasse che circa *s\*α* (in questo caso 13) statistiche fallissero. Per questo motivo possiamo concludere che il generatore in questione supera positivamente il test di uniformità.

