

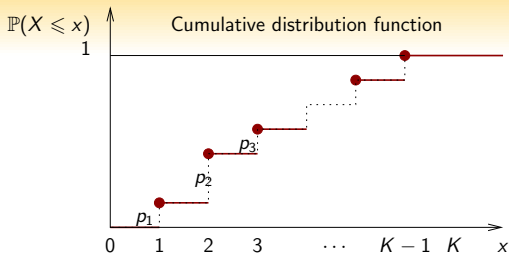
# Random generation of Discrete Random variables

Jean-Marc Vincent

MESCAL-INRIA Project  
Laboratoire d'Informatique de Grenoble  
Universities of Grenoble, France  
{Jean-Marc.Vincent}@imag.fr



# Inverse of PDF



## Generation

Divide  $[0, 1[$  in intervals with length  $p_k$   
 Find the interval in which *Random* falls  
 Returns the index of the interval  
 Computation cost :  $\mathcal{O}(\mathbb{E}X)$  steps  
 Memory cost :  $\mathcal{O}(1)$

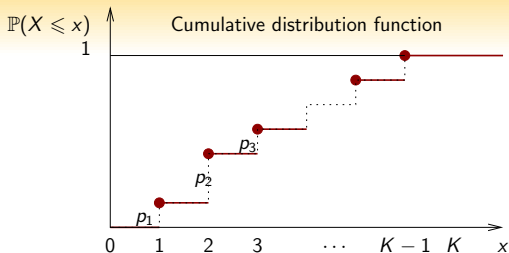
## Inverse function algorithm

```

s=0; k=0;
u=random()
while u > s do
  k=k+1
  s=s+pk
end while
return k
  
```



# Inverse of PDF



## Generation

Divide  $[0, 1[$  in intervals with length  $p_k$   
 Find the interval in which *Random* falls  
 Returns the index of the interval  
 Computation cost :  $\mathcal{O}(\mathbb{E}X)$  steps  
 Memory cost :  $\mathcal{O}(1)$

## Inverse function algorithm

```

s=0; k=0;
u=random()
while u > s do
  k=k+1
  s=s+pk
end while
return k
  
```

# Searching optimization

## Optimization methods

- pre-compute the pdf in a table
- rank objects by decreasing probability
- use a dichotomy algorithm
- use a tree searching algorithm (optimality = Huffmann coding tree)

## Comments

- Depends on the usage of the generator (repeated use or not)
- pre-computation usually  $\mathcal{O}(K)$  could be huge
-

# Searching optimization

## Optimization methods

- pre-compute the pdf in a table
- rank objects by decreasing probability
- use a dichotomy algorithm
- use a tree searching algorithm (optimality = Huffmann coding tree)

## Comments

- Depends on the usage of the generator (repeated use or not)
- pre-computation usually  $\mathcal{O}(K)$  could be huge
-

# Aliasing technique

## Initialization

$K$  objects

list  $L = \emptyset, U = \emptyset;$

**for**  $k=1; k \leq K; k++$  **do**

$P[k] = p_k$

**if**  $P[k] \geq \frac{1}{K}$  **then**

$U = U + \{k\};$

**else**

$L = L + \{k\};$

**end if**

**end for**

## Alias and threshold tables

**while**  $L \neq \emptyset$  **do**

Extract  $k \in L$

Extract  $i \in U$

$S[k] = P[k]$

$A[k] = i$

$P[i] = P[i] - (\frac{1}{K} - P[k])$

**if**  $P[i] \geq \frac{1}{K}$  **then**

$U = U + \{i\};$

**else**

$L = L + \{i\};$

**end if**

**end while**

Combine uniform and alias value when rejection



# Aliasing technique

## Initialization

$K$  objects

list  $L = \emptyset, U = \emptyset;$

**for**  $k=1; k \leq K; k++$  **do**

$P[k] = p_k$

**if**  $P[k] \geq \frac{1}{K}$  **then**

$U = U + \{k\};$

**else**

$L = L + \{k\};$

**end if**

**end for**

## Alias and threshold tables

**while**  $L \neq \emptyset$  **do**

Extract  $k \in L$

Extract  $i \in U$

$S[k] = P[k]$

$A[k] = i$

$P[i] = P[i] - (\frac{1}{K} - P[k])$

**if**  $P[i] \geq \frac{1}{K}$  **then**

$U = U + \{i\};$

**else**

$L = L + \{i\};$

**end if**

**end while**

Combine uniform and alias value when rejection

# Aliasing technique

## Initialization

$K$  objects

list  $L = \emptyset, U = \emptyset;$

**for**  $k=1; k \leq K; k++$  **do**

$P[k] = p_k$

**if**  $P[k] \geq \frac{1}{K}$  **then**

$U = U + \{k\};$

**else**

$L = L + \{k\};$

**end if**

**end for**

## Alias and threshold tables

**while**  $L \neq \emptyset$  **do**

Extract  $k \in L$

Extract  $i \in U$

$S[k] = P[k]$

$A[k] = i$

$P[i] = P[i] - (\frac{1}{K} - P[k])$

**if**  $P[i] \geq \frac{1}{K}$  **then**

$U = U + \{i\};$

**else**

$L = L + \{i\};$

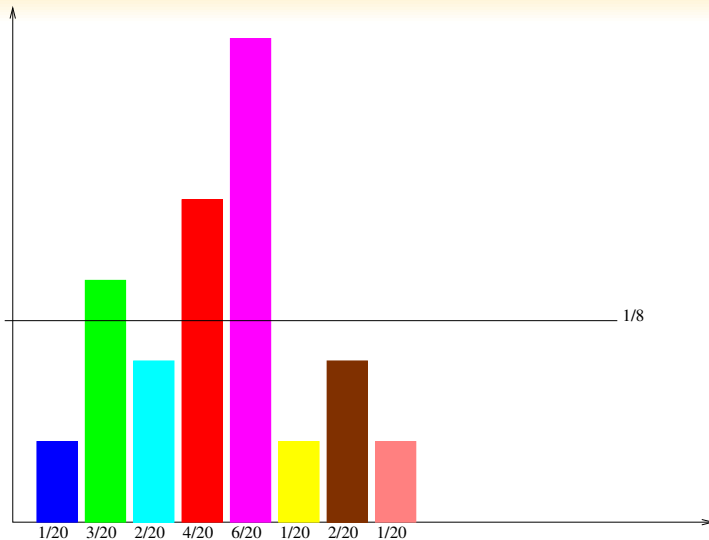
**end if**

**end while**

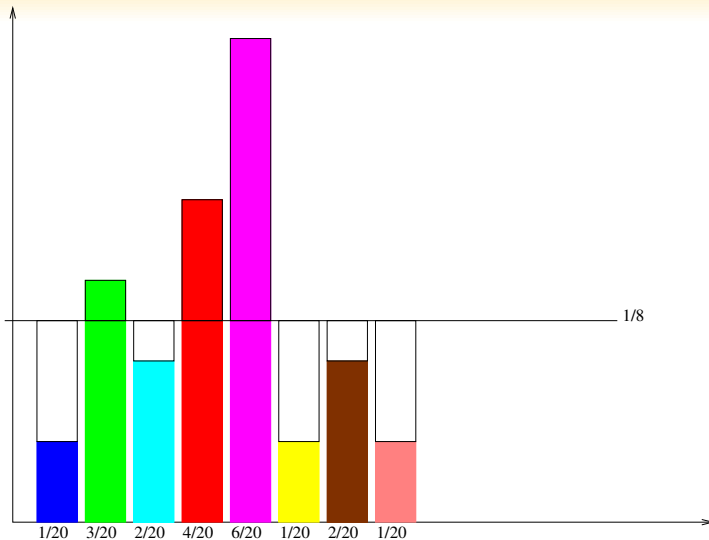
Combine uniform and alias value when rejection



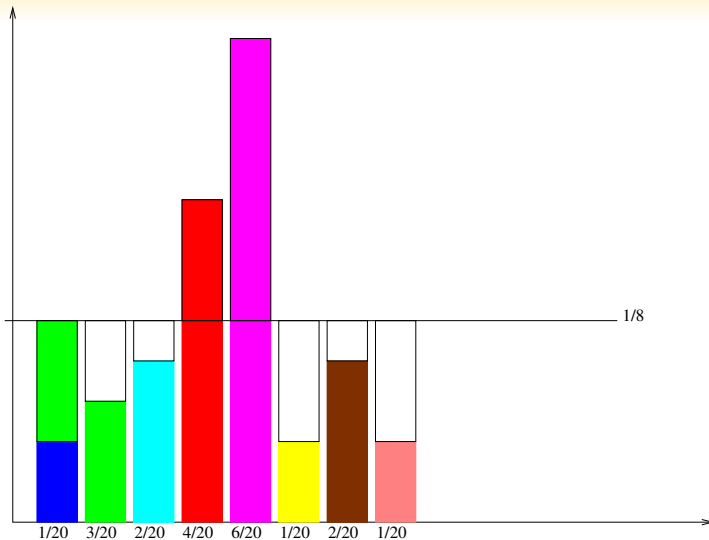
# Aliasing technique : generation



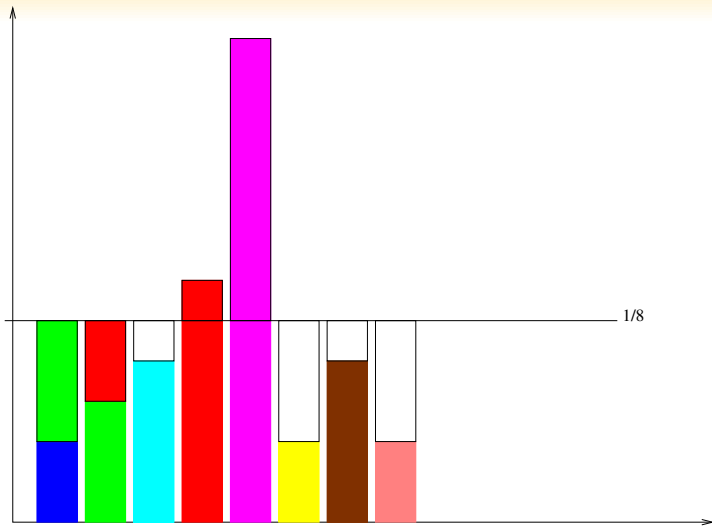
# Aliasing technique : generation



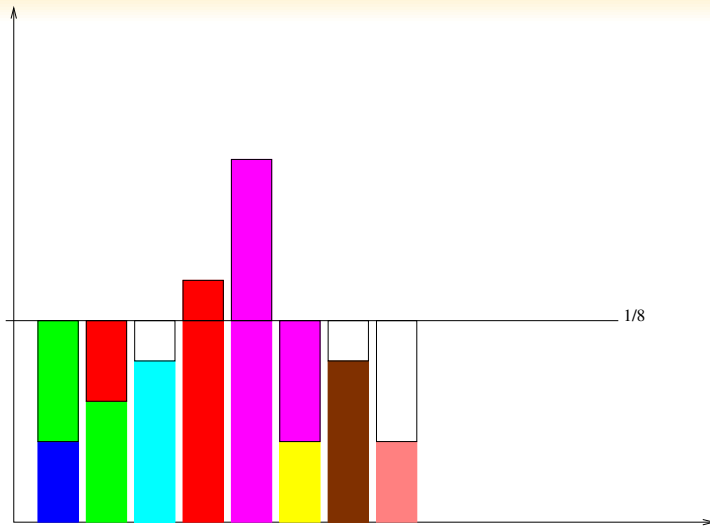
# Aliasing technique : generation



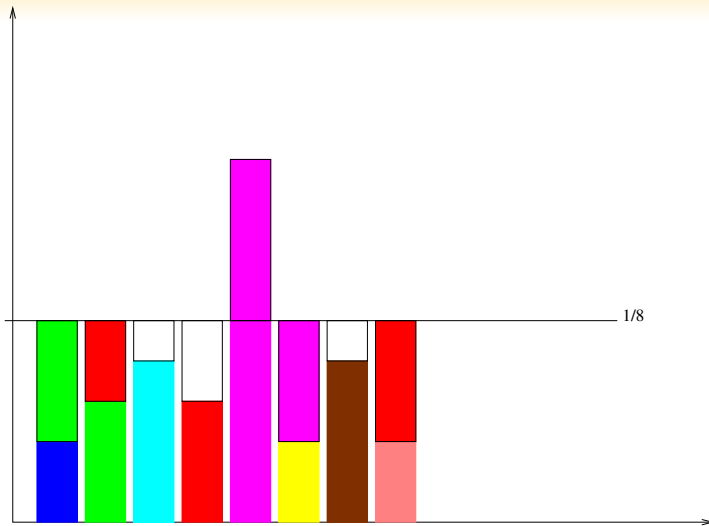
# Aliasing technique : generation



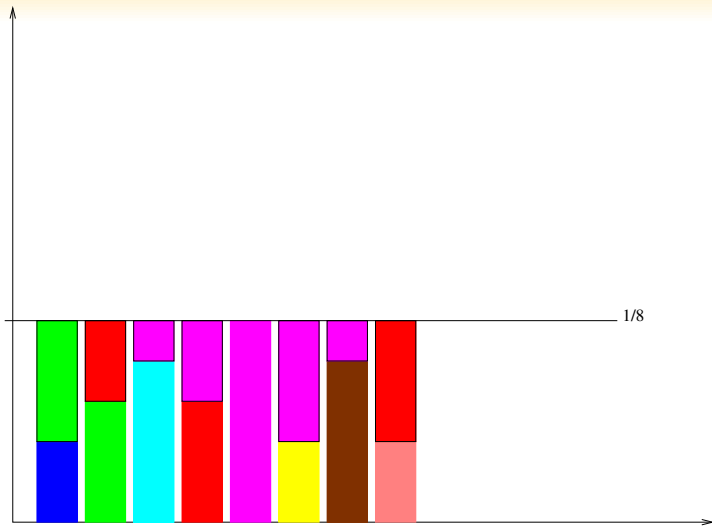
# Aliasing technique : generation



# Aliasing technique : generation



# Aliasing technique : generation



# Aliasing technique : generation

## Generation

```

k=alea(K)
if Random .  $\frac{1}{K} \leq S[k]$  then
  return k
else
  return A[k]
end if

```

## Complexity

Computation time :

- $\mathcal{O}(K)$  for pre-computation
- $\mathcal{O}(1)$  for generation

Memory :

- threshold  $\mathcal{O}(K)$  (real numbers as probability)
- alias  $\mathcal{O}(K)$  (integers indexes in a tables)



# Aliasing technique : generation

## Generation

```

k=alea(K)
if Random .  $\frac{1}{K} \leq S[k]$  then
  return k
else
  return A[k]
end if

```

## Complexity

Computation time :

- $\mathcal{O}(K)$  for pre-computation
- $\mathcal{O}(1)$  for generation

Memory :

- threshold  $\mathcal{O}(K)$  (real numbers as probability)
- alias  $\mathcal{O}(K)$  (integers indexes in a tables)