Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

Continuous Random Variates: applications

Università degli studi di Roma Tor Vergata
Department of Civil Engineering and Computer Science Engineering

1

For our application framework, we will look at:

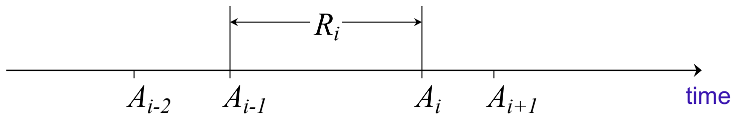
- arrival process model
- service process model

2

L'istante di arrivo i-esimo è la somma dei primi 'i' tempi di INTERARRIVO, che sono positivi ovviamente, allora la costruzione dei tempi di ARRIVO è crescente.

Arrival process model

- Model *interarrival* times as RV sequence $R_1, R_2, R_3, \ldots$
- Construct corresponding arrival times $A_1, A_2, A_3, \ldots$ defined by
  $A_0 = 0$ and $A_i = A_{i-1} + R_i \quad i=1, 2, \ldots$
- by induction, $A_i = R_1 + R_2 + \ldots + R_i \quad i=1, 2, \ldots$
- since $R_i > 0$, $0 = A_0 < A_1 < A_2 < A_3 < \ldots$



Example

- programs `ssq2` and `ssq3` generate job arrivals in this way, where R_1, $R_2, R_3, \ldots$ are *Exponential*($1/\lambda$).
  In both programs, the arrival rate is equal to $\lambda = 0.5$ jobs per unit time

```
 double GetArrival()
{ static double arrival = START;
  SelectStream(0);
  arrival += Exponential(2.0);
  return (arrival);}
```

Implementazione della slide precedente, ovvero Ai = somma degli interarrivi exp.

Fino a sis2 modellavamo quantità richiesta in settimana, uniformemente spalmata nella settimana, domanda unitaria.
Nella pratica, questi sono limiti, e con sis3 e sis4 abbiamo modellato il caso per togliere questi limiti.
Con sis3 siamo passati da generare la quantità di richiesta degli articoli per ogni intervallo di tempo con una distribuzione.
In sis3 avevamo domanda media 30, con sis4 abbiamo tolto il vincolo di '1 richiesta = 1 ordine di prodotto', riadattando quindi lambda per far si che la media tornasse sempre 30. Abbiamo modellato ciò con una geometrica.

Example

- programs `sis3` and `sis4` generate demand instances in this way, with *Exponential*($1/\lambda$) interdemand times.
  The demand rate corresponds to an average of
  - $\lambda = 30.00$ actual demands per time interval in `sis3`
  - $\lambda = 120.00$ potential demands per time interval in `sis4`

```
double GetDemand(long *amount)
/* ------------------------------------------------
 * generate a demand instance with rate 120
 * and generate a corresponding demand amount
 * ------------------------------------------------
 */
{
  static double time = START;

  SelectStream(0);
  time += Exponential(1.0 / 120.0);      /* demand instance */
  SelectStream(2);
  *amount = Geometric(0.2);               /* demand amount   */
  return (time);
}
```

```
double GetDemand(void)
/* ---------------------------------------------------------------
 * generate the next demand instance (time) with rate 30 per time
 * interval and exactly one unit of demand per demand instance
 * ---------------------------------------------------------------
 */
{
  static double time = START;

  SelectStream(0);
  time += Exponential(1.0 / 30.0);
  return (time);
}
```


5

Se ho interarrivi indipendenti e i.d., la sequenza di arrivi A1,.., è un processo stazionario di arrivi di tasso lambda. Anche detti processi di rinnovo o processi omogenei.

Def stationary arrival process

If $R_1, R_2, R_3, \ldots$ is an *iid* sequence of positive interarrival times with $E[R_i] = 1/\lambda > 0$, then the corresponding sequence of arrival times $A_1, A_2, A_3, \ldots$ is a *stationary arrival process* with rate λ

- also known as
  - *Renewal* processes
  - *Homogeneous* arrival processes
- arrival rate λ has units "arrivals per unit time" tempo = 1/frequenza
  - If average interarrival time is 0.1 minutes,
  - then the arrival rate is 10 arrivals per minute
- stationary arrival processes are "convenient fiction"

  important theoretically
  good approximation
  understand for nonstationary
- If the arrival rate λ varies with time, the arrival process is *nonstationary*

6

Nella realtà, è difficile dire che la 'media' non cambi nel tempo, basti pensare al problema delle fasce orarie.

Se il tasso lambda cambia nel tempo,
il processo di arrivo NON E' STAZIONARIO.

Si parte dallo stazionario per poter capire il 'non' stazionario. Se nel progetto avessi un nonstazionario, posso dividere in fasce orarie, associando una media ciascuna.

Senza troppe informazioni su arrivi, spesso si assume Exp(1/lambda), se prendiamo lambda come tasso, nel parametro c'è la media.

stationary Poisson arrival process

random events!

- As in `ssq2`, `ssq3`, `sis3`, `sis4`, with lack of information it is usually most appropriate to assume that the interarrival times are *Exponential*(1/λ)

- If $R_1, R_2, R_3, \ldots$ is an *iid* sequence of *Exponential*(1/λ) interarrival times, the corresponding sequence $A_1, A_2, A_3, \ldots$ of arrival times is a stationary *Poisson* arrival process with rate λ
  Equivalently, for $i=1, 2, 3, \ldots$, the arrival time A_i is an *Erlang*(i,1/λ) random variable

$$A_i = R_1 + R_2 + \ldots + R_i$$

7

Algorithm 1

Given $\lambda > 0$ and $t > 0$, this algorithm generates a *realization* of a stationary Posson arrival process with rate λ over $(0, t)$

```
a0 = 0.0;              /* a convention */
n = 0;
while(an < t) {
    an+1 = an + Exponential(1 / λ);
    n++;
}
return a1, a2, a3, . . . , an;
```

Per convenzione il tempo inizia ad istante 0, metto poi gli arrivi n = 0, genero tutti i tempi di interarrivo dall'esponenziale, poi conto gli arrivi, questa è realizzazione del processo stocastico di arrivo di Poisson.

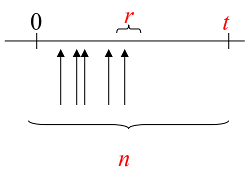
8

Prendo intervallo di tempo da '0' a 't', al suo interno ci metto un certo numero di arrivi 'n', e un sottointervallo posto in modo random rispetto all'intervallo (0,t).

Random Arrivals

We now demonstrate the interrelation between Uniform, Exponential and Poisson random variables.

Consider:
- $t > 0$, defines a fixed time interval $(0, t)$
- n rappresents the number of arrivals in the interval $(0, t)$
- $r > 0$, is the length of a small subinterval located at random interior to $(0, t)$

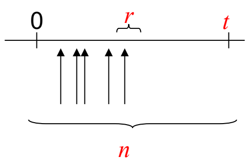


Correspondingly:
- $\lambda = n / t$ is the arrival rate
- $p = r / t$ is the probability that a particular arrival will be in the subinterval
- $np = nr / t = \lambda r$ is the expected number of arrivals in the subinterval

lambda*r = tasso * finestra temporale

9

random arrivals → Poisson



Theorem 1

Let:
- $A_1, A_2, A_3, \ldots$ be an iid sequence of *Uniform*(0, t) random variables (“unsorted” arrivals).
- the discrete random variable X be the number of A_i that fall in a fixed subinterval of length $r = pt$ interior to $(0, t)$

If n is large and r / t small, X is indistinguishable from a *Poisson*(λr) random variable with $\lambda = n / t$

media attesa
come parametro

10

Conclusions on random arrivals

- if *many* arrivals occur *at random* with a rate of λ, the number of arrivals X that will occurr in an interval of length r is *Poisson(λr)*
- The probability of x arrivals in an interval with length r is

$$Pr(X = x) = \frac{e^{-\lambda r}(\lambda r)^x}{x!} \qquad x = 0,1,2,..$$

- The probability of <u>no arrivals</u> is: $Pr(X = 0) = e^{-\lambda r}$
- The probability of <u>at least one arrival</u> is:

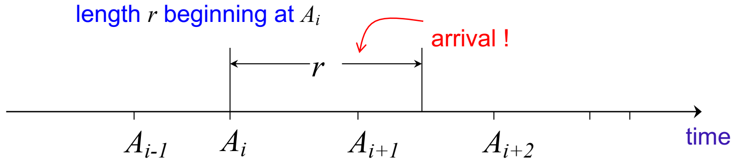
$$Pr(X > 0) = 1 - Pr(X = 0) = 1 - e^{-\lambda r}$$

For a fixed λ, the probability of at least one arrival increases with increasing interval length r

Random Arrivals $\rightarrow$ Exponential Interarrivals

R interarrivo
- If R represents the time between consecutive arrivals, the possible values of R are $r > 0$
- Consider arrival time A_i selected at random and an interval of length r beginning at A_i

in 'r' vedo quanti arrivi cadono, nella figura cade solo A(i+1), Ai è fuori dal range



- $R = A_{i+1} - A_i$ will be less than r iff there is at least one arrival in this interval
- the cdf of R is
  $Pr(R \le r) = Pr$ (at least one arrival in r)=1- $e^{-\lambda r}$
- R is an *Exponential(1/λ)* random variable

Dagli arrivi random discende l'interarrivo esponenziale.

Theorem 2

If arrivals occur at random with rate λ, the corresponding interarrival times form an iid sequence of *Exponential(1/λ)* RVs.

This result justifies the use of *Exponential* interarrival times in programs `ssq2`, `ssq3`, `sis3`, `sis4`

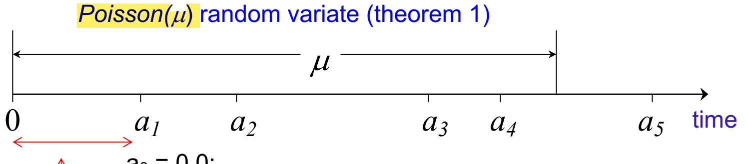
- If we know only that arrivals occur at random with a constant rate λ, the function `GetArrival` in `ssq2` and `ssq3` is appropriate
- If we know only that demand instances occur at random with a constant rate λ, the function `GetDemand` in `sis3` and `sis4` is appropriate

Supponiamo lambda = 1

Generating Poisson random *variates*

Observation:

- If arrivals occur at random with rate $\lambda=1$
- the number of arrivals X in an interval of length μ will be a *Poisson(μ)* random variate (theorem 1)



exponential

```
a0 = 0.0;
x = 0;
while (a < μ) {
    a += Exponential(1.0);
    x++;
}
return x-1;
```

$a_5 > \mu$

x-1 perchè partiamo da 0 e facciamo l'ultimo incremento finchè sono dentro 'mu'.

Summary of Poisson arrival processes

Given a fixed time interval $(0, t)$, there are two ways of generating a realization of a stationary Poisson arrival process with rate λ

come generare la Poisson:

1. Generate the number of arrivals: n = *Poisson*(λt)
   Generate a *Uniform*(0,t) random variate sample of size n and sort to form $0 < a_1 < a_2 < a_3 < \ldots < a_n$

2. use algorithm 1 with *Exponential*($1/\lambda t$)

genero tempi interarrivo esponenziali, e poi genero processo Poisson

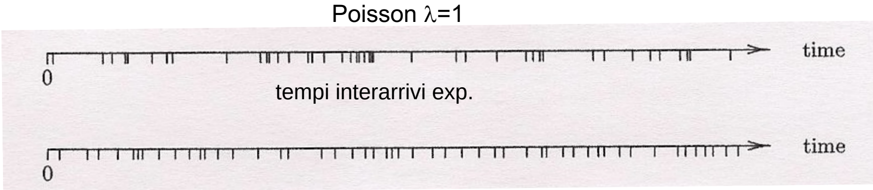
- Statistically, the two approaches are equivalent
- The first approach is computationally more expensive, especially for large n
- The second approach is always preferred

15

Summary of Poisson arrival processes

- The *mode* of the exponential distribution is 0
  $\rightarrow$ A stationary Poisson arrival process exhibits "clustering"

Poisson λ=1



Interarrival times *Erlang*(4,1/4)

nel primo caso ho alcuni arrivi raggruppati, sotto è più 'uniforme'.

The stationary Poisson arrival process generalizes to

- a stationary arrival process when exponential interarrival times are replaced by any continuous RV with positive support
- a nonstationary Poisson arrival process when λ varies over time

16

Per le continue, abbiamo visto come esempi di applicazione il caso degli arrivi: arrivi random, tempi interarrivi exp, numero arrivi poissoniani.

Processi di servizio

Qui le considerazioni sono molto più varie. Il processo dei servizi presenta situazioni variegate, esistono lineeguida.

For our application framework, we will look at:

- arrival process model
- service process model

Service Process Models

differently from the case of arrival processes, there are no well-defined "default", only application-dependent guidelines:

- *Uniform*(a, b) service times are usually inappropriate since they rarely "cut off" at a maximum value b

La Normale prende anche valori negativi, o non la uso o la tronco.

- Service times are positive, so they cannot be *Normal*(μ,σ) unless truncated to positive values
- Positive probability models "with tails", such as the *Lognormal*(a, b) distribution, are candidates
  anche Pareto!

• jobs UNIX
• web file size
• Internet topology
• IP packet flow
• …

- If service times are the sum of n iid *Exponential*(b) sub-task times, then the *Erlang*(n, b) model is appropriate

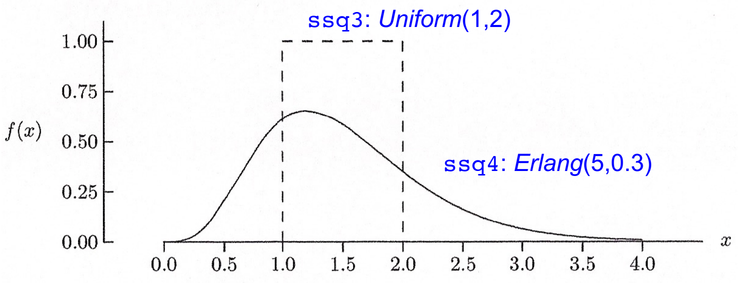
Non esiste regola generale, negli arrivi stavo tranquillo con l'Esponenziale per la maggior parte dei casi. Sui servizi devo capire bene cosa sia adatto.

Program ssq4 (ssq3 aveva Uniform)

- ssq4 is based on program ssq3, but with a more realistic *Erlang*(5, 0.3) service time model 5*0.3= 1.5, come ssq3
  The corresponding service rate is 2/3
- As in program ssq3, ssq4 uses *Exponential*(2) random variate interarrivals.
  The corresponding arrival rate is 1/2 frequenza è 0.5

Example

- For both ssq3 and ssq4, the arrival rate is λ=0.5 and the service rate is $\nu = 2/3 \approx 0.667$
- The distribution of service times for two programs is very different a parità di media



first conjecture how the steady-state statistics will differ for these two programs and then investigate the correctness of your conjecture via discrete-event simulation

Che differenza di prestazioni si ha tra Uniform ed Erlang?

Example

- suppose using a *Normal*(1.5,2.0) random variable to model service times
- Truncate distribution so that
  - Service times are non-negative $(a=0)$
  - Service times are less than 4 $(b=4)$

sia Uniform che Erlang hanno tempi tra 0 e 4.

come troncare? conosco i punti.

```
α = cdfNormal(1.5, 2.0, a);          /* a is 0.0 */
β = 1.0 - cdfNormal(1.5, 2.0, b);    /* b is 4.0 */
```

- the result: $\alpha = 0.2266$ and $\beta = 0.1056$

uso le cumulative per calcolare le 'probabilità' date i punti 0 e 4, definendo le due probabilità di coda

NB
the *truncated Normal*(1.5,2.0) random variable has a mean of 1.85 (not 1.5) and a standard deviation of 1.07 (not 2.0)

Why is the mean increased?????

21

E' ovvio perchè ho tolto dei 'pezzi', quindi tutto è più concentrato verso il 'centro'.

Come posso generare queste variabili variate troncate?
cioè per modellare questa distribuzione troncata?

Constrained Inversion

Noi usiamo l'inversione vincolata, ovvero: calcate alfa e beta, parto dalla probabilità in quel range.
Genero 0<u<1 per costruire la variata, la metto già nell'insieme troncato.

once α and β are determined, the corresponding truncated random variate can be generated by using constrained inversion

```
u = Uniform(α, 1.0 - β);
return F⁻¹(u);
```

Exercise:
- generate n truncated random variates;
- compute sample mean and standard deviation (by Welford)
- verify for which n the statistics converge to the theoretical values

(quelli scritti nella slide precedente)

22

Example (cont.)

- The idf capability in `rvms` can be used to generate the *truncated Normal*(1.5,2.0) RV

α = 0.2266274;
β = 0.1056498;
u = Uniform(α, 1.0 - β);
return idfNormal(1.5, 2.0, u);

genero valore tra alfa e 1-beta, questo u generato è u= 0.7090, l'inversa della normale ritorna x = 2.601

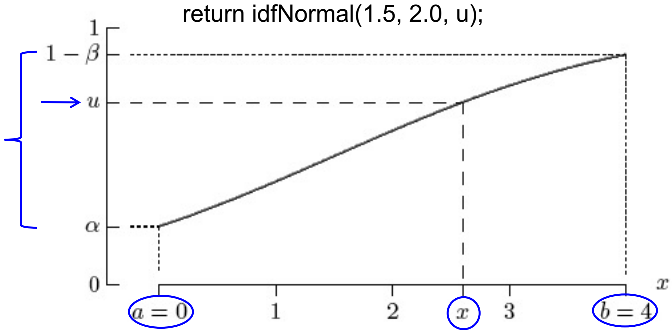


Figure shows u=0.7090 and x=2.601

Exercises

- Exercise 7.3.1