HYPENS: un simulatore per le reti di Petri discrete, continue e ibride.

Fausto Sessego

Dip. di Ing. Elettrica ed Elettronica, Università di Cagliari, Piazza d'Armi, 09123 Cagliari, Italy
email:sessegofausto@tiscali.it

16 Luglio 2007

Ringrazio i miei genitori che mi hanno dato la possibilità di raggiungere questo traguardo e tutti coloro che mi sono stati vicini.

Ringrazio, inoltre, i professori Alessandro Giua, Carla Seatzu e Elio Usai che mi hanno seguito durante lo svolgimento della tesi.

Indice

Capitolo 1

Introduzione

Le reti di Petri sono un modello di sistemi ad eventi discreti che trae origine dal lavoro di Carl Adam Petri, un ricercatore tedesco, che nel 1962 discusse la sua tesi di dottorato dal titolo 'Kommunication mit Automaten', in cui presentava questo nuovo modello matematico che descriveva le relazioni esistenti fra *condizioni* ed *eventi*. Esse rappresentano uno degli strumenti più efficaci nell'analisi di tutto quel ramo dell'automatica che si occupa di studiare i sistemi dinamici ad eventi discreti (SED). L'analisi e l'ottimizzazione di modelli a eventi discreti richiede un grande sforzo computazionale: infatti i sistemi industriali sono sono difficili da analizzare e da controllare. I problemi di scala reale spesso arrivano ad essere intrattabili sia analiticamente che computazionalmente; per far fronte a questo problema sono stati sviluppati e applicati i modelli fluidi i quali sono una approssimazione a dinamica continua dei modelli discreti. Notiamo che, per descrivere lo stesso sistema di produzione, sono necessarie differenti approssimazioni fluide che dipendono dal suo stato discreto: la macchina è in funzione oppure è ferma, i buffers sono pieni oppure vuoti e così via. Il modello risultante può essere descritto attraverso un modello ibrido dove ad ogni stato discreto è associato una diversa dinamica. Le reti di Petri (PN) sono state introdotte, inizialmente, per descrivere ed analizzare sistemi ad eventi discreti; recentemente ci si è maggiormente indirizzati ad applicare questi modelli a sistemi ibridi.

In questa tesi verrà presentato un simulatore in grado di simulare l'evoluzione nel tempo delle reti di Petri discrete temporizzate (TPN), reti di Petri continue con transizioni a velocità massima costante (CPN) e reti di Petri ibride (HPN) date dall'unione di reti di Petri discrete temporizzate e reti di Petri continue.

Nel capitolo 2 introdurremo la notazione e i concetti relativi alle TPN, la politica di evoluzione guidata da una semantica di servente finito o infinito, i tipi di transizione temporizzata deterministica e stocastica; grazie alle reti TPN con transizioni stocastiche ad andamento esponenziale abbiamo potuto analizzare problemi quali le reti di code markoviane. Si è curato in particolar modo il pro-

blema relativo al conflitto generale con il relativo metodo di risoluzione adottato seguendo l'approccio proposto in [1]. Tutta la trattazione sarà accompagnata da degli esempi.

Nel capitolo 3 tratteremo la notazione e i concetti relativi alle CPN, esplicando il significato di posti e transizioni continue, spiegando cosa vuol dire associare una velocità a ciascuna transizione; sarà curato, anche nel caso delle CPN, il problema relativo al conflitto effettivo con il relativo metodo di risoluzione adottato. Inoltre introdurremo il concetto di ottimizzazione dell' evoluzione della rete in relazione a una funzione obiettivo; un esempio, trattato all'interno del capitolo, chiarirà i concetti elencati sopra.

Nel capitolo 4 parleremo delle HPN, che sono costituite dall'unione di reti di Petri temporizzate e reti di Petri continue; verrà mostrato come valgono tutti i concetti introdotti nei capitoli precedenti e la stretta dipendenza che vige fra la parte discreta e la parte continua. Le HPN sono un potentissimo mezzo di analisi per sistemi complessi: tutta la trattazione sarà resa più comprensibile grazie all'ausilio di un semplice esempio relativo a un sistema di produzione.

Il capitolo 5 è il fulcro della tesi: esso si occupa del simulatore il grado di simulare ed analizzare tutti i tipi di rete quali TPN, CPN, HPN. Il simulatore è costituito da 4 file: 2 servono per l'interfaccia, 1 è utilizzato per la simulazione e il quarto serve per analizzare i dati ottenuti.

Il capitolo 6 riguarda le simulazioni vere e proprie con i rispettivi risultati che verranno accostati a quelli teorici: si potrà notare l'elevata efficienza del simulatore e l'utilità del quarto file (quello relativo all'analisi dei risultati) per lo studio dei risultati ottenuti e la loro possibilità di visulaizzazione grafica. Verranno simultate tutte le reti trattate in modo teorico nei capitoli precedenti.

Il capotolo 7 è quello conclusivo.
I contributi che questa tesi ha portato alla ricerca sono di natura teorica ed implementativa (sviluppo di software in MATLAB):

- Nelle reti di Petri temporizzate discrete si sono introdotti algoritmi per la risoluzione di conflitto generale che si basano sull'associazione, a ciascuna transizione, di un peso e di un livello di priorità.

- Nelle reti di Petri temporizzate continue si sono introdotti i concetti di velocità minima e massima associate alle transizioni.

- Si è sviluppato un toolbox in grado di simulare reti di Petri discrete, continue e ibride, assegnando un tempo massimo di simulazione ed eventualmente (in caso ci siano transizioni continue) la funzione obiettivo.

- Il simulatore è stato testato, in particolare, nel caso di reti di Petri limitate e stocastiche, che come è ben noto, sono in grado di descrivere reti complesse

di code markoviane. Si è trovato che i valori forniti dal simulatore per tempi di simulazione sufficientemente lunghi tendono ai valori ideali forniti dalle tabelle.

Ci sono, infine, due appendici: l'appendice A tratta in modo dettagliato le possibili funzioni di distribuzione che possono assumere le transizioni continue mentre l'appendice B mostra il codice sorgente del file $simulator\_HPN.m$.

Capitolo 2

Reti di Petri discrete

2.1 Introduzione alle reti di Petri discrete

In questo capitolo viene prima descritto un modello di rete di Petri più semplice, detto *rete posto/transizione* o PN, poi verranno presentate le reti di Petri *temporizzate*, deterministiche e stocastiche, che indicheremo con TPN; per le TPN saranno introdotti e concetti di grado di abilitazione, semantica di servente, orologi delle transizioni e conflitti fra transizioni. Fra i vari modelli ad eventi discreti, le reti di Petri hanno un'importanza predominante a causa di vari fattori:

- Le PN sono un formalismo *grafico* e *matematico* allo stesso tempo. Essendo un formalismo grafico sono di facile comprensione, possono essere usate come strumento di aiuto visuale in fase di progetto e specifica, consentono al progettista e all'operatore di un sistema di parlare lo stesso linguaggio; essendo un formalismo matematico consentono di applicare una vasta gamma di tecniche di analisi per studiare le proprietà di interesse.

- Permettono di dare una *rappresentazione compatta* di sistemi con grande spazio di stato; infatti non richiedono di rappresentare esplicitamente tutti i valori possibili dello stato di un sistema, ma solo le regole che governano la sua evoluzione.

- Permettono di rappresentare esplicitamente il concetto di *concorrenza*, cioè di attività che possono venire svolte parallelamente.

- Consentono una *rappresentazione modulare*, cioè se un sistema è composto da più sottosistemi che interagiscono fra loro, è generalmente possibile rappresentare ciascun sottosistema come una semplice sottorete e poi, mediante operatori di rete, unire le varie sottoreti per ottenere il modello del sistema complessivo.

Le reti di Petri si dividono in logiche e temporizzate: le reti di Petri logiche o PN, non consentono di rappresentare la temporizzazione degli eventi, ma solo l'ordine con cui essi si verificano; le reti di Petri temporizzate o TPN si dividono a loro volta in deterministiche e stocastiche. Inoltre le reti di Petri possono distinguersi in *autonome* oppure *non autonome*: le prime descrivono l'evoluzione di un sistema i cui istanti di scatto sono sconosciuti o non indicati mentre le seconde (che sono quelle che andremo a considerare) mostrano il funzionamento di un sistema la cui evoluzione è condizionata da eventi esterni (*sincronizzate*) e/o dal tempo (*temporizzate*). Noi tratteremo reti di Petri temporizzate non autonome.

2.1.1 Reti di Petri posto/transizione

Una rete di Petri P/T è un grafo bipartito, orientato e pesato. I due tipi di vertici sono detti posti (rappresentati da cerchi) e transizioni (rappresentati da barre o da rettangoli). Gli archi, che devono essere orientati, connettono i posti alle transizioni e viceversa.

Definizione 2.1.1. *Una PN è una struttura $N_d = (P, T, Pre, Post)$ dove:*

- $P = \{p_1, p_2, \cdots, p_{md}\}$ *è un insieme finito di* m_d posti *non vuoto.*
- $T = \{t_1, t_2, \cdots, t_{nd}\}$ *è un insieme finito di* n_d transizioni *non vuoto.*
- $Pre : P \times T \longrightarrow \mathbb{N}$*: è la* funzione di pre-incidenza *che specifica gli archi diretti dai posti alle transizioni (detti archi Pre) e viene rappresentata mediante una matrice $m_d \times n_d$; più precisamente, $Pre(p_i, t_j)$ indica quanti archi vanno dal posto p_i alla transizione t_j.*
- $Post : P \times T \longrightarrow \mathbb{N}$*: è* funzione di post-incidenza *che specifica gli archi diretti dalle transizioni ai posti (detti archi $Post$) e viene rappresentata mediante una matrice $m_d \times n_d$; più precisamente, $Post(p_i, t_j)$ indica quanti archi vanno dalla transizione t_j al posto p_i .*

Si noti che in una rete di Petri discreta $P_d = P$ e $T_d = T$, cioè l'insieme di tutti i posti e transizioni della rete sono discreti. Le matrici Pre e $Post$ sono delle matrici di interi non negativi. Si denota con $Pre(\cdot, t_j)$ la colonna della matrice Pre relativa alla transizione t_j e con $Pre(p_i, \cdot)$ la riga della matrice Pre relativa al posto p_i. La stessa notazione vale per la matrice $Post$. Si suppone che $P \cap T = \emptyset$, cioè posti e transizioni sono insiemi disgiunti e che $P \cup T \neq \emptyset$, cioè la rete è costituita da almeno un posto o da una transizione. L'informazione sulla struttura di rete contenuta nelle matrici Pre e $Post$ può essere compattata in un'unica matrice, detta di incidenza.

Definizione 2.1.2. *Data una rete N_d, con m_d posti ed n_d transizioni, la* matrice di incidenza $C : P \times T \longrightarrow \mathbb{Z}$ *è la matrice $m_d \times n_d$ definita come:*

$$C = Post - Pre$$

cioè il generico elemento di C vale $C(p_i, t_j) = Post(p_i, t_j) - Pre(p_i, t_j)$.

Data C non posso ricostruire il grafo, mentre date le matrici Pre e $Post$ posso ricostruire perfettamente il grafo. Un esempio chiarirà questi concetti.

Esempio 2.1.1. *In Figura 2.1 è rappresentata la rete $N_d = (P, T, Pre, Post)$ con insieme dei posti $P = \{p_1, p_2, p_3, p_4\}$ e l'insieme delle transizioni $T = \{t_1, t_2, t_3\}$.*

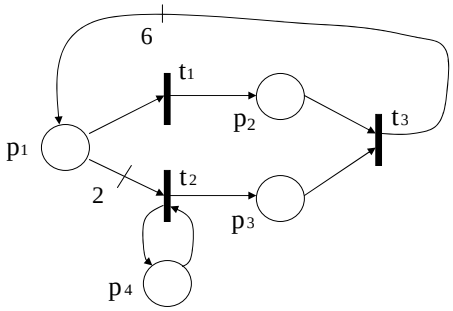


Figura 2.1: Una rete di Petri posto-transizione.

Le matrici Pre e $Post$ valgono:

$$Pre = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad Post = \begin{bmatrix} 0 & 0 & 6 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

La matrice di incidenza vale:

$$C = \begin{bmatrix} -1 & -2 & 6 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Si noti come non sia possibile risalire al cappio tra la transizione t_2 e il posto p_4 a partire dalla matrice di incidenza.

Data una transizione si definiscono i seguenti insiemi di posti:

- Insieme dei posti in ingresso alla transizione t_j

$$^\bullet t_j = \{p_i \in P \mid Pre(p_i, t_j) > 0\}.$$

- Insieme dei posti in uscita dalla transizione t_j

$$t_j^\bullet = \{p_i \in P \mid Post(p_i, t_j) > 0\}.$$

Dato un posto si definiscono i seguenti insiemi di transizioni:

- Insieme delle transizioni in ingresso al posto p_i

$$^\bullet p_i = \{t_j \in T \mid Post(p_i, t_j) > 0\}.$$

- Insieme delle transizioni in uscita dal posto p_i

$$p_i^\bullet = \{t_j \in T \mid Pre(p_i, t_j) > 0\}.$$

Ad esempio nella rete in Figura 2.1 vale $^\bullet t_3 = \{p_2, p_3\}$, $t_2^\bullet = \{p_3, p_4\}$, $^\bullet p_2 = \{t_1\}$, $p_2^\bullet = \{t_3\}$.

2.1.2 Marcatura e sistema di rete

Mediante la marcatura è possibile definire lo stato di una PN.

Definizione 2.1.3. *Una* marcatura *è una funzione $M : P \longrightarrow \mathbb{N}$ che assegna ad ogni posto un numero intero non negativo di marche (o gettoni) rappresentate graficamente con dei pallini neri dentro i posti.*

Definizione 2.1.4. *Una rete N_d con una marcatura iniziale M_0 è detta* rete marcata o sistema di rete *e viene indicata come $R_d = \langle N_d, M_0 \rangle$.*

Si noti che il vettore della generica marcatura M è un vettore colonna $m_d \times 1$,quindi ha tante righe quanti sono i posti della rete. Una rete marcata è, in effetti, un sistema ad eventi discreti a cui è associato un comportamento dinamico.

Esempio 2.1.2. *Considerando l'esempio in Figura 2.2, la marcatura iniziale M_0 è $M_0(p_1) = 5, M_0(p_2) = 0, M_0(p_3) = 0, M_0(p_4) = 1$.*

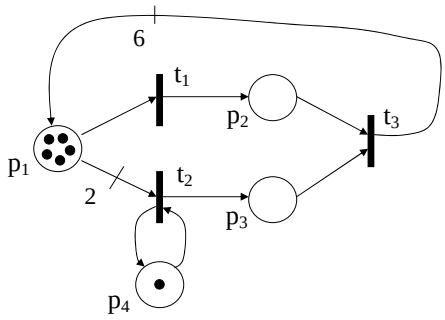


Figura 2.2: Una rete di Petri marcata posto-transizione.

2.1.3 Abilitazione e scatto

Definizione 2.1.5. *Una* transizione t_j *è* abilitata *dalla marcatura* M *se*

$$M \geq Pre(\cdot, t_j)$$

cioè se per ogni posto $p_i \in P$ *della rete contiene un numero di marche pari o superiore a* $Pre(p_i, t_j)$*. Una transizione* t_j *abilitata da una marcatura* M *può scattare: lo scatto di* t_j *rimuove* $Pre(p_i, t_j)$ *marche da ogni posto* $p_i \in P$ *e aggiunge* $Post(p_i, t_j)$ *marche in ogni posto* $p_i \in P$*, determinando una nuova marcatura* M'*. Cioè vale:*

$$M' = M - Pre(\cdot, t_j) + Post(\cdot, t_j) = M + C(\cdot, t_j).$$

Per indicare che lo scatto di t_j *da* M *determina la marcatura* $M^{'}$ *si scrive* $M[t_j\rangle M^{'}$*.*

Esempio 2.1.3. *Ad esempio, consideriamo la Figura 2.2: posso decidere di far scattare* t_1 *e* t_2 *in quanto sono abilitate. Ad esempio, posso far scattare la transizione* t_1 *per* 5 *volte consecutive oppure la transizione* t_2 *per* 2 *volte consecutive; si noti che posso decidere anche di far scattare, ad esempio,* t_1 *una volta e poi* t_2 2 *volte, ottenendo, in quest'ultimo caso, un'evoluzione della rete come mostrato in Figura 2.3, caratterizzata da un nuovo valore della marcatura* M' *pari a* $M^{'}(p_1) = 0, M^{'}(p_2) = 1, M^{'}(p_3) = 2, M^{'}(p_4) = 1.$

Definizione 2.1.6. *Una sequenza di transizioni* $\sigma = t_x\, t_y \cdots t_z \in T$ *è abilitata da una marcatura* M *se la transizione* t_x *è abilitata da* M *e il suo scatto porta a* $M_1 = M + C(\cdot, t_1)$*, se la transizione* t_y *è abilitata da* M_1 *e il suo scatto porta a* $M_2 = M_1 + C(\cdot, t_y)$*, etc.*

Una sequenza abilitata σ viene anche detta sequenza di scatto alla quale corrisponde la traiettoria:

$$M[t_x\rangle M_1[t_y\rangle M_2 \cdots [t_z\rangle M_r.$$

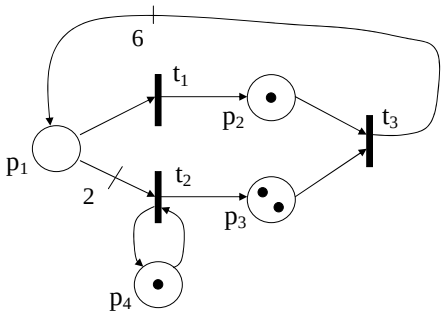


Figura 2.3: Evoluzione della rete marcata in Figura **??**. Marcatura raggiunta dopo lo scatto di t_1,t_2 e t_2

Per indicare che la sequenza σ è abilitata da M si scrive $M[\sigma\rangle$. Per indicare che lo scatto di σ, a partire dalla marcature M, determina la marcatura M' si scrive$M[\sigma\rangle M'$. Ad esempio, nella rete in Figura 2.2, una possibile sequenza di scatto, a partire dalla marcatura M, è $\sigma = t_2t_1t_2$, il cui scatto porta alla marcatura $M' = [\ 0 \quad 1 \quad 2 \quad 1\ ]^T$ come mostrato in Figura 2.3.

Definizione 2.1.7. *Data una rete $\langle N_d, M_0\rangle$ con un insieme di transizioni $T = \{t_1, t_2, \cdots, t_{nd}\}$ ed una sequenza di transizioni $\sigma \in T$, si definisce* vettore caratteristico di σ*, il vettore di scatto $\vec{\sigma}$ la cui generica componente $\vec{\sigma_j} = |\sigma|_{tj}$ indica quante volte la transizione t_j compare in σ.*

In una rete marcata N_d, avente una matrice di incidenza C, se M è raggiungibile da M_0 attraverso la sequenza di transizioni σ, allora possiamo scrivere che

$$M = M_0 + C \cdot \vec{\sigma}.$$

dove $\vec{\sigma}$ è il vettore di scatto associato alla sequenza di transizioni σ.

2.2 Reti di Petri temporizzate

Nel paragrafo precedente è stato evidenziato come le reti di Petri siano uno strumento potente per lo studio di caratteristiche strutturali e qualitative dei sistemi a eventi discreti (SED). E' altrettanto evidente che, qualora si desideri analizzare le prestazioni di un SED, cioè effettuarne uno studio quantitativo, le reti posto/transizione non sono uno strumento adatto, perchè prescindono dalla durata delle singole attività che avvengono nel sistema. Si è dunque concluso che, per rendere le reti di Petri uno strumento utile per la valutazione delle prestazioni, deve essere introdotta in esse una qualche metrica temporale: da qui nascono le

reti di Petri temporizzate (TPN). Come strumento modellistico dei SED, esse offrono molti vantaggi che possono essere sintetizzati nei punti che seguono:

- Le TPN forniscono un ambiente idoneo ad una corretta e dettagliata rappresentazione di un SED, consentendo anche di definire le stocasticità intrinseche dei sistemi reali; esse infatti, come le reti posto/transizione, sono regolate da semantiche ben definite che, oltre che costituire la base per i metodi di analisi formale, consentono di realizzare simulatori TPN.
- Le TPN possono riprodurre esattamente caratteristiche proprie dei SED quali priorità, sincronizzazione, uso di risorse condivise;
- L'analisi di TPN può essere effettuata in modo automatico mediante l'uso di strumenti informatici dedicati;
- Le TPN possono essere utilizzate nello stesso tempo come modelli logici e prestazionali.

I concetti e i formalismi trattati in tutto il capitolo fanno riferimento a [2] e [3].

2.2.1 Temporizzazione e concetti base

Sappiamo che nelle reti di Petri autonome una transizione può scattare se questa è abilitata, ma non abbiamo ancora specificato quando questa può scattare. Associamo un tempo d_j, di valore zero se non è specificato, alla transizione t_j.

Definizione 2.2.1. *Una rete di Petri temporizzata è data dalla coppia* $\langle R, Tempo \rangle$ *dove:*

- R_d *è una* PN *marcata;*
- *Tempo:* $T \longrightarrow \mathbb{R}_0^+$ *è una funzione associata all'insieme di transizioni T secondo la quale* $Tempo(t_j) = d_j$ *= tempo associato a* t_j*;*

Lo scatto viene considerato un'operazione indivisibile e la temporizzazione è realizzata associando a ogni transizione un ritardo che corrisponde a tempo che deve trascorrere fra la sua abilitazione ed il conseguente scatto; i gettoni rimangono nei posti in ingresso a ciascuna transizione fino allo scatto, a meno che un altra transizione le prelevi a sua volta. Solo se, al termine del ritardo d_j, la condizione di abilitazione continua a sussistere (senza che durante questo intervallo venga mai persa), la transizione t_j scatta effettivamente facendo sì che i gettoni siano prelevati dai posti in ingresso e quindi depositati nei posti in uscita. Indichiamo con τ il tempo totale che è trascorso a partire dall'istante 0. Consideriamo la Figura 2.4: la transizione t_1, avente un ritardo d_1, è abilitata al tempo $\tau = \tau_1$, e può

scattare al tempo $\tau = \tau_1 + d_1$: stiamo, cioè, associando alla transizione temporizzata abilitata al tempo $\tau = \tau_1$ un clock di durata d_j che entra in funzione a partire dal tempo τ_1. Se, durante il lasso di tempo $\tau_1 \leq \tau \leq \tau_1 + d_1$, la transizione t_1 venisse disabilitata, automaticamente perderebbe il clock associatogli all'istante $\tau = \tau_1$ e quindi non potrebbe più scattare al tempo previsto $\tau = \tau_1 + d_1$.

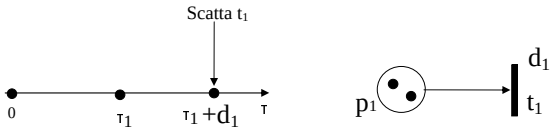


Figura 2.4: Evoluzione temporale della transizione t_1 abilitata al tempo τ_1.

Definizione 2.2.2. *Una transizione t_j di una TPN è detta:*

- immediata*, se scatta appena viene abilitata, o, in altre parole, se a essa è associato un tempo di ritardo nullo;*
- deterministica*, se a essa è associato un tempo di ritardo d_j scelto in modo deterministico;*
- stocastica*, se il tempo di ritardo d_j a essa associato è una variabile aleatoria con funzione distribuzione di probabilità nota; si consulti l'appendice A per maggiori approfondimenti.*

Per poter studiare le TPN dobbiamo introdurre 2 concetti fondamentali che analizzeremo nel paragrafo successivo:

1. Nella definizione 2.1.5 si è parlato del concetto di abilitazione: ora, nelle TPN, parleremo di *grado di abilitazione* associato ad ogni transizione;
2. Spiegheremo cosa vuol dire associare un *numero di serventi* ad ogni transizione.

2.2.2 Grado di abilitazione e numero di serventi

Definizione 2.2.3. *Una* transizione t_j *è* abilitata *dalla marcatura M se*

$$M \geq Pre(\cdot, t_j)$$

cioè se per ogni posto $p_i \in P$ della rete contiene un numero di marche pari o superiore a $Pre(p_i, t_j)$.

Quindi continua ancora a valere la definizione data per le reti posto/transizione. Per le TPN si introduce un nuovo concetto; il *grado di abilitazione* associato ad ogni transizione.

Definizione 2.2.4. *Data una marcatura M si definisce grado di abilitazione della transizione t_j*

$$enab(M, t_j) = \max\{k \in \mathbb{N} : \quad M \geq k\ Pre(\cdot, t_j)\}$$

allora t_j è detta essere $enab(M, t_j)$ volte abilitata.

Questo vuol dire che la transizione t_j, avente un ritardo d_j a partire dall'istante di tempo $\tau = \tau_j$, potrebbe scattare $enab(M, t_j)$ volte all'istante $\tau = \tau_j + d_j$. Possiamo definire un nuovo vettore riga e_{ab} che avrà m_d colonne (pari al numero delle transizioni), dove $e_{ab}(j) = enab(M, t_j)$ indicherà il grado di abilitazione per la transizione corrispondente all' indice della colonna. Quindi $e_{ab}(j)$ sarà il grado di abilitazione della transizione t_j. Consideriamo la Figura 2.5. La transizione

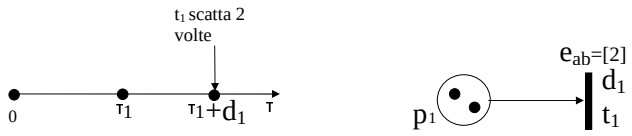


Figura 2.5: Evoluzione temporale della transizione t_1 abilitata al tempo τ_1 con grado di abilitazione 2.

t_1 (ha un grado di abilitazione pari a 2 a partire dall'istante di tempo $\tau = \tau_1$; quindi possiamo scrivere che $e = [2]$, cioè che t_1 può scattare 2 volte all'istante $\tau = \tau_1 + d_1$.

Consideriamo ora l'esempio in Figura 2.6. Le transizioni t_1 e t_2 hanno ritardo rispettivamente pari a d_1 e d_2 dove $d_2 > d_1$. Entrambe le transizioni hanno grado di abilitazione pari a 1 al tempo $\tau = 0$: la transizione t_1 può scattare dopo un tempo pari a $\tau = d_1$ mentre la transizione t_2 può scattare dopo un tempo pari a $\tau = d_2$. Dopo un tempo $\tau = \tau_1$ scatta la transizione t_1 la quale toglie un gettone dal posto p_1 e ne mette uno nel posto p_2: al tempo $\tau = \tau_1$ avrò 2 gettoni nel posto p_2. Questo non fa altro che aumentare da 1 a 2 il grado di abilitazione di t_2 cioè non fa altro che aumentare da 1 a 2 il numero di orologi associati alla transizione t_2. Un orologio associato a t_2 è quello che è partito dall'istante $\tau = 0$ e che permetterà lo scatto di un gettone dopo un tempo pari a $\Delta_1 = d_2 - d_1$ detto *tempo residuo di scatto*; l'altro orologio partirà dall'istante $\tau = d_1$ e permetterà lo scatto del secondo gettone dopo un tempo pari a $\Delta_2 = d_2$. Il concetto importante è che a ogni transizione posso associare uno o più orologi in funzione al grado di abilitazione che essa assume.

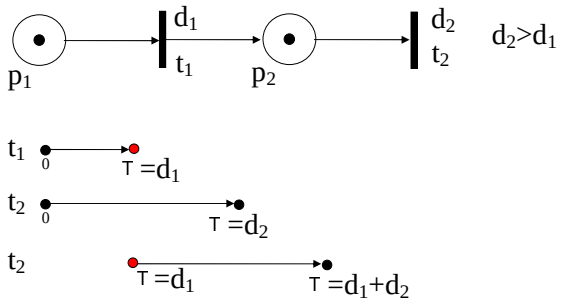


Figura 2.6: Evoluzione temporale delle transizioni t_1 e t_2 con i rispettivi orologi associati a partite dal tempo $\tau = 0$.

Negli esempi visti fino ad ora, abbiamo sotto inteso implicitamente una semantica di serventi infiniti; ma che cosa rappresentano i serventi? Un servente è un'unità operativa in grado di *servire* la transizione alla quale è associato e rimane occupato fino a quando non ha terminato il *servizio*. Risulta evidente che se più serventi sono associati ad una stessa transizione, possono lavorare in parallelo e la transizione sarà *servita* in maniera più veloce; è chiaro come ci sia uno stretto legame fra il numero di serventi associati ad una transizione e il grado di abilitazione che essa possiede. Facciamo un esempio per chiarire il concetto.

Esempio 2.2.1. *Assumiamo che la transizione t_1 in Figura 2.7 modelli una operazione di assemblaggio: un componente, corrispondente a un gettone nel posto p_1, è assemblato con un componente corrispondente a un gettone nel posto p_2: il prodotto risultante corrisponde a un gettone nel posto p_3. La transizione t_1, in base alla definizione 2.2.4, è 2 volte abilitata in quanto $e_{ab} = [min(2, 3)] = [2]$. Ma è possibile rendere contemporaneamente disponibili 2 pezzi assemblati? Assumiamo che ci sia un solo servente (può essere una sola macchina o un*

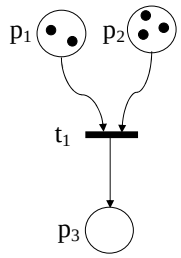


Figura 2.7: TPN generica: t_1 è 2 volte abilitata.

solo operatore): la sua rappresentazione è differente se consideriamo il caso di limitazione implicita *oppure* limitazione esplicita*:*

Limitazione implicita: *a ogni transizione associamo uno ed un solo servente. Per avere x serventi che* servono *una transizione bisogna aumentare di x il numero delle transizioni, ciascuna delle quali avrà in ingresso e in uscita gli stessi posti (con lo stesso numero di gettoni) e gli stessi archi (con lo stesso peso) della transizione originaria.Questo provocherà un aumento di x colonne uguali delle matrici Pre e $Post$ (e quindi anche della matrice C) per ogni transizione che si vuole servire con x serventi;*

Limitazione esplicita: *il numero di serventi di ogni transizione è uguale al numero di gettoni presenti in un posto collegato alla transizione stessa attraverso un cappio (chiaramente tale definizione vale solo nel caso in cui il peso degli archi del cappio sia unitario).*

Ridefiniamo la rete in Figura 2.7 a seconda del caso di limitazione implicita *e* limitazione esplicita*.*

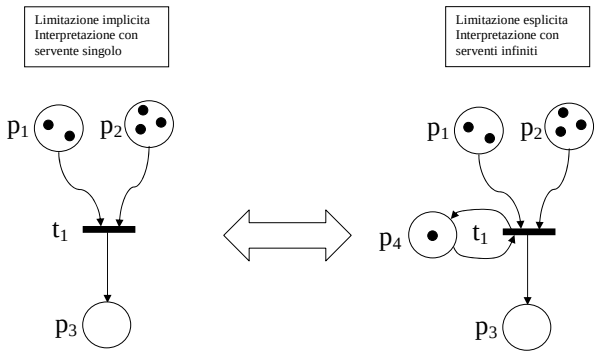


Figura 2.8: TPN con 1 solo servente disponibile: limitazione implicita con interpretazione a servente singolo e corrispondente limitazione esplicita con interpretazione a serventi infiniti.

Considerando la Figura 2.8, rappresentiamo il singolo servente con la singola transizione t_1 nel caso di limitazione implicita *e con con un posto p_4 con marcatura unitaria nel caso di* limitazione esplicita*: ci chiediamo se il servente singolo è sufficiente a servire la transizione t_1 avente in origine (Figura 2.7) grado di abilitazione 2: la risposta è negativa. Infatti nonostante il grado di abilitazione della transizione t_1 fosse in origine pari a 2 (cioè si avevano 2 coppie di pezzi da assemblare), poichè non ho* fisicamente 2 *serventi che possono lavorare le coppie di pezzi in parallelo (ogni servente assembla una coppia di pezzi), il servente singolo sarà costretto ad assemblare prima una coppia e poi,dopo che l'ha assemblata e l'ha tolta dal ciclo di assemblamento mettendola nel posto p_3, potrà prendere in consegna la successiva coppia. Tutto questo equivale a dire che*

la transizione t_1 può scattare allo stesso tempo solo una volta. Risulta evidente una qualsiasi transizione può essere abilitata k-volte ma non è detto che scatti $k - volte$ contemporaneamente: la frequenza di scatto è limitata dal numero di serventi che sono associati alla transizione stessa e che sono disponibili; se una transizione è k-volte abilitata e possiede un numero di serventi pari a x, la frequenza di scatto per quella transizione sarà $min(k, x)$. Considerando la Figura 2.8 nel caso di limitazione implicita*, il grado di abilitazione della transizione t_1 è $e_{ab}(1) = min(2, 3) = 2$ se non specifichiamo che la transizione t_1 è servita da un solo servente! Se specifichiamo che t_1 ha un solo servente, il vettore di scatto di t_1 varrà $e_{ab}(1) = min(2, 3, 1)) = 1$. Quindi la formula matematica che ci permette di calcolare il grado di abilitazione di una transizione non risulta completa per il caso di* limitazione implicita *a servente singolo mentre lo è per il caso di* limitazione esplicita*. In base alla definizione di* limitazione implicita *e di* limitazione esplicita*, se avessimo voluto associare alla transizione t_1, della TPN in Figura 2.7, un numero di serventi pari a 2, avremmo avuto 2 TPN equivalenti come quelle rappresentate in Figura 2.9.*

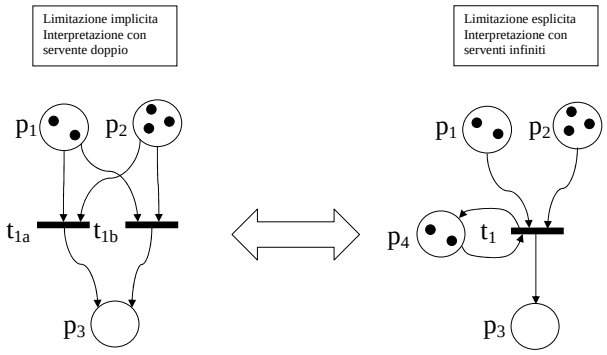


Figura 2.9: TPN con 2 serventi disponibili: limitazione implicita con interpretazione a doppio servente e corrispondente limitazione esplicita con interpretazione a serventi infiniti.

Si vede come, nel caso di *limitazione implicita*, si sia dovuta sdoppiare la transizione t_1 in t_{1a} e t_{1b} le quali, lavorando in parallelo, simulano la presenza di un doppio servente associato alla transizione t_1 originaria; se calcoliamo la matrice di incidenza per la TPN originaria (Figura 2.7) e per la nuova TPN nel caso di *limitazione implicita* (Figura 2.9), otteniamo rispettivamente:

$$C = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} -1 & -1 \\ -1 & -1 \\ 1 & 1 \end{bmatrix}.$$

Come si era già detto, per aumentare da 1 a x il numero di serventi associati alla stessa transizione nel caso di *limitazione implicita*, si deve prendere la transizione in questione e replicarla nella TPN per x volte mantenendo le stesse connessioni con i posti; cioè equivale ad aggiungere, nella matrice di incidenza C, un numero di colonne uguali pari a r. Quindi all' aumentare del numero di serventi aumenta a dismisura anche il numero di transizioni nella rete rendendola più complessa e meno leggibile; per questo motivo si preferisce adottare la *limitazione esplicita* che ci permette, nell'esempio in questione, di assegnare un numero di serventi alla transizione t_1 tramite l'aggiunta del posto p_4. D'ora in avanti considereremo reti di Petri con *limitazione esplicita* a serventi infiniti ad eccezione dei casi in cui non specificheremo il contrario.

La rappresentazione grafica e numerica può essere caotica anche se si utilizza la *limitazione esplicita*: basti pensare ad una rete con n_d transizioni ciascuna delle quali ha un cappio per indicarne i serventi; in questo modo avrei x posti in più in tutta la rete cioè avrei x righe in più nelle matrici Pre,$Post$,C e M. Per ovviare a questo problema, si è introdotto un vettore dei serventi s:

Definizione 2.2.5. *Data una rete di Petri R_d, costruiamo un vettore s di dimensione $1 \times n_d$ memorizzando il numero di serventi $s_j \in s$ associato a ogni transizione $t_j \in T$ nella colonna j-esima.*

Quindi calcoliamo le matrici Pre,$Post$,C non considerando al loro interno i posti che sono collegati alle transizioni tramite un cappio; quest'ultima informazione la andiamo a considerare all'interno del vettore s dove memorizziamo il numero di serventi associati alle transizioni. Se una transizione t_j non ha cappio vuol dire che ha un numero di serventi infiniti che la può servire: in quest'ultimo caso, all'interno del vettore s, associerò in corrispondenza della colonna j-esima il valore ∞. E' chiaro che, una volta che ci siamo costruiti il vettore s, è inutile raffigurare i cappi indicanti i serventi associati alle transizioni in quanto sarebbe una informazione che abbiamo già all'interno di s. Le TPN in Figura 2.10 sono equivalenti e mi forniscono la stessa evoluzione

Se una transizione ha un grado di abilitazione pari a k, posso definire un vettore dei tempi residui di scatto contente k componenti ciascuno dei quali sarà un certo orologio partito ad un determinato istante di tempo. Per ogni transizione posso fare lo stesso ragionamento: ciò mi permette di costruire una **matrice degli orologi** Q che sarà costituita dalla concatenazione su più righe di tutti i vettori dei tempi residui di scatto associati alle transizioni della TPN. Tali orologi vengono inizializzati ogni volta che la transizione viene abilitata ai valori specificati dalla struttura di temporizzazione; in particolare, nel caso deterministico, sono impostati a valori costanti. Se ad una stessa transizione sono associati più orologi, questi vengono memorizzati nella matrice Q in ordine crescente, da sinistra verso destra. Bisogna porre attenzione sul dimensionamento della matrice Q che

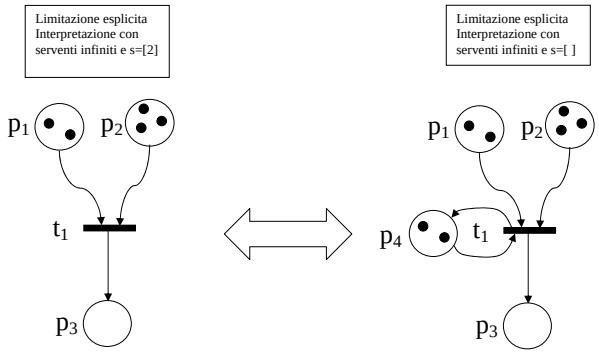


Figura 2.10: TPN con 2 serventi disponibili: limitazione esplicita con il supporto del vettore s e corrispondente limitazione esplicita con il vettore s vuoto.

dipende strettamente dalle dimensioni dei vettori dei tempi residui di scatto i quali a loro volta dipendono dal vettore di abilitazione: poichè durante l'evoluzione di una rete TPN le transizioni possono assumere diversi valori del grado di abilitazione, questo vuol dire che il vettore di abilitazione e_{ab} avrà al suo interno dei valori diversi tra di loro e variabili nel tempo. Poichè il valore $e_{ab}(j)$ ci fornisce il grado di abilitazione di t_j e quindi il numero di elementi che possiede il vettore dei tempi residui di scatto associato a t_j, ci possono essere vettori dei tempi di scatto assocciati alle diverse transizioni che possono avere una dimensione differente; nel momento in cui andiamo a fare la concatenazione dei vettori dei tempi residui di scatto per costruire la matrice degli orologi Q, possono esserci elementi della matrice Q vuoti. A questi elementi associamo come valore NaN cioè Not-a-Number perchè non rappresentano niente, ma servono solamente per il corretto riempimento e dimensionamento della matrice Q; è importante notare che il numero massimo di colonne di Q sarà dato dal numero massimo di colonne di e_{ab}. Usualmente, la marcatura iniziale è tale per cui i i tempi residui di scatto sono massimi, quindi ci riferiamo sempre a $M_{0,max}$ se non è specificato il contrario.

Si vuole porre l'attenzione su un caso particolare: consideriamo la TPN in Figura 2.11.

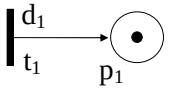


Figura 2.11: Una transizione senza archi in ingresso.

Una transizione che non possiede archi in ingresso, come la transizione t_1, è detta *transizione sorgente*. Una transizione sorgente è sempre abilitata, poichè,

essendo in tal caso $Pre(\cdot, t_1) = 0$, la condizione di abilitazione è sempre verificata: inoltre, poichè il grado di abilitazione è infinito, la transizione t_1 scatta infinite volte per $\tau = d_1$. D'ora in avanti eviteremo questa situazione, imponendo che

$$\begin{cases} Pre(\cdot, t_1) > 0, & \text{se} \quad s(t_1) = [0, \infty] \\ Pre(\cdot, t_1) = 0, & \text{se} \quad s(t_1) = [0, \infty[\end{cases}$$

cioè imponendo che il grado di abilitazione di una transizione sia limitato o dai posti in ingresso oppure dal numero di serventi associati alla transizione stessa.

Algoritmo 2.2.1. ***Evoluzione temporale di una rete di Petri temporizzata*** *Se, a partire da una marcatura M_l, t_j ha un grado di abilitazione pari a $k = e_{ab}(j) = enab(M, t_j)$, si ha che $Q(t_j, \cdot) = [o_{j,1}, \cdots, o_{j,k}]$. Detto $o_j = \min\ Q(t_j, \cdot)$ il più piccolo dei valori degli orologi associati alla transizione t_j, esso rappresenta il tempo che deve passare, a partire dall'istante attuale, prima del prossimo scatto di t_j, se t_j è attualmente abilitata e non viene disabilitata prima che trascorra l'intervallo o_i. L'evoluzione della marcatura di una TPN avviene attraverso la ripetizione dei seguenti passi:*

1. *Inizializzare l'indice $l = 0$.*
2. *Identificare lo stato corrente della TPN contraddistinto dalla marcatura corrente M_l.*
3. *Calcolare il vettore dei serventi s.*
4. *Inizializzare la matrice degli orologi $Q = \emptyset$.*
5. *Calcolare il grado di abilitazione $enab(M_l, t_j)$ di ogni transizione t_j a partire dalla marcatura M_l (definizione 2.2.4) e costruire il vettore di abilitazione $e_{ab,l}$.*
6. *Calcolare il grado di abilitazione di ogni transizione abilitata in funzione del numero di serventi associati alle transizioni stesse*

   $$e_{ab,l}(j) = \min_{(} s(j), e_{ab,l}(j)) \qquad t_j \in T$$

   Se $\exists e_{ab,l}(j) > 0, \quad t_j \in T$ allora vai al passo successivo altrimenti vai al passo 12)
7. $l = 0$**:** *Se $t_j \in T$ ha un grado di abilitazione pari a $k = e_{ab,l}(j) = enab(M_l, t_j)$, si ha che $Q(t_j, \cdot) = o_{j,1}, \cdots, o_{j,k}$, altrimenti detto $z = \max(e_{ab,l})$ poniamo $Q(t_j, \cdot) = NaN_{j,1}, \cdots, NaN_{j,z}$*

$l \neq 0$ *(a) Se il grado di abilitazione* $e_{ab,l}(j)$ *associato alla transizione* t_j *a partire dalla marcatura* M_l *è inferiore al grado di abilitazione* $e_{ab,l-1}(j)$ *associato alla transizione* t_j *a partire dalla marcatura* M_{l-1}*, allora devono essere scartati* $[e_{ab,l-1}(j) - e_{ab,l}(j)]$ *orologi associati alla transizione* t_j*, che risultano evidentemente in esubero: vengono selezionati ed eliminati dalla riga* j*-esima della matrice* Q *gli* $[e_{ab,l-1}(j) - e_{ab,l}(j)]$ *orologi che hanno valori più alti.*

(b) Se, invece, $e_{ab,l}(j) > e_{ab,l-1}(j)$*,* $[e_{ab,l}(j) - e_{ab,l-1}(j)]$ *orologi devono essere associati a* t_j *e impostati ai valori specificati dalla struttura di temporizzazione* $Tempo$*.*

8. *Individuare* o^**, mediante la relazione*

$$o^* = \min_{j:t_j \in T} \{o_i\}$$

come il valore più piccolo fra gli orologi o_j *associati alle transizioni* t_j *abilitate nella marcatura* M_l*; la transizione corrispondente a* $o_j = o^*$ *è denotata come* t_j^* *e memorizzata nel vettore di scatto* $\vec{\sigma}$*. Se il minimo non è unico, ma* o^* *è pari al valore minimo di orologio di diverse transizioni, queste si trovano a scattare contemporaneamente, qualora non vi siano dei conflitti generali e la sequenza di scatto viene memorizzata nel vettore di scatto* $\vec{\sigma}$*. In caso contrario bisogna adottare una delle tecniche di risoluzione dei conflitti, come spiegato nel paragrafo 2.2.4*

9. *Il tempo avanza fino all'istante* $\tau_{l+1} = \tau_l + o^*$ *e il sistema evolve giungendo alla nuova marcatura* $M_{l+1}(\tau_{l+1}) = M_l(\tau_l) + C \cdot \vec{\sigma}$

10. *Incrementare l'indice* $l = l + 1$ *associato allo stato del sistema.*

11. *Ritorna al passo* $5)$

12. *Fine dell'evoluzione totale della* TPN*.*

Quando facciamo evolvere la rete si ha un cambiamento nel tempo, dovuto allo scatto delle transizioni dopo un ritardo d associato, della marcatura M, del vettore di abilitazione e_{ab}, della matrice degli orologi Q; possiamo mantenere tutte queste informazioni all'interno del cosiddetto *grafo di raggiungibilità*.

Esempio 2.2.2. *consideriamo la rete in Figura 2.12.*

Analizziamo la rete nel dettaglio per un tempo di evoluzione pari a $\tau = 8$*. Le matrici* Pre *e* $Post$ *valgono:*

$$Pre = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad Post = \begin{bmatrix} 0 & 0 & 6 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

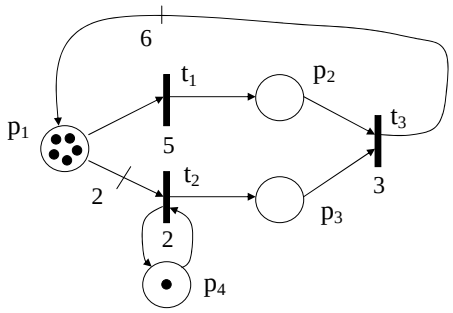


Figura 2.12: Rappresentazione di una TPN con transizioni deterministiche.

La matrice di incidenza vale:

$$C = \begin{bmatrix} -1 & -2 & 6 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}.$$

Ci si può chiedere perchè nelle matrici Pre,$Post$ e C non abbiamo considerato il posto p_4: la risposta è che consideriamo p_4 come un servente singolo, quindi stiamo riducendo la complessità delle matrici Pre,$Post$ e C a fronte di introdurre il vettore dei serventi $s = (\infty, 1, \infty)$, dove il valore in $s(1) = s(3) = \infty$ sta ad indicare che le transizioni t_1 e t_3 hanno un numero di serventi infiniti che le possono servire; analogamente il valore $s(2) = 1$ sta ad indicare che soltanto un servente può processare la transizione t_2. La marcatura iniziale (che consideriamo a partire dal tempo $\tau = 0$) vale:

$$M_0 = [M_0(p_1), M_0(p_2), M_0(p_3)]^T = [5, 0, 0]^T$$

Per questa marcatura, il vettore di abilitazione $e_{ab,0}$ avrà il seguente valore:

$$e_{ab,0} = [q(t_1, M_0), q(t_2, M_0), q(t_3, M_0)] = [5, 1, 0]$$

La matrice degli orologi Q_0 avrà la seguente forma:

$$Q_0 = \begin{bmatrix} 5 & 5 & 5 & 5 & 5 \\ 2 & NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN & NaN \end{bmatrix}$$

In Figura 2.12, per questioni di leggibilità, abbiamo omesso tutti gli NaN (Not-a-Number) relativi alla matrice Q; si può subito notare l'influenza del servente singolo sulla transizione t_2 la quale, altrimenti, avrebbe avuto un grado di abilitazione pari a 2 e quindi 2 orologi anzicchè 1. Quindi, a partire dalla marcatura

iniziale M_0, si ha la transizione t_1 con grado di abilitazione pari a 5 e la transizione t_2 con grado di abilitazione pari a 1: a partire dal tempo $\tau = 0$ associo 5 orologi, ciascuno con tempo pari a $d_1 = 5$, alla transizione t_1, e poi associo anche 1 orologio con tempo $d_2 = 2$, alla transizione t_1. La transizione che scatta prima è quella che ha l'orologio più basso e cioè t_2 dopo un tempo di clock $\tau = 2$ che prende 2 gettoni dal posto p_1 e mette un gettone nel posto p_3; quindi, in seguito allo scatto di t_2, dobbiamo aggiornare il vettore M, e_{ab} e la matrice Q. La marcatura M_1 raggiunta dopo un tempo $\tau_1 = 2$ ed un tempo totale di evoluzione pari a $\tau = 2$vale

$$M_1 = [M_1(p_1), M_1(p_2), M_1(p_3)]^T = [3, 0, 1]^T$$

Per questa marcatura, il vettore di abilitazione $e_{ab,1}$ avrà il seguente valore

$$e_{ab,1} = [q(T_1, M_1), q(T_2, M_1), q(T_3, M_1)] = [3, 1, 0]$$

Lo scatto di t_2 ha fatto abbassare il grado di abilitazione di t_1 ma non ha disabilitato la transizione: t_1 avrà, pertanto, ancora degli orologi associati all'interno della matrice Q_1 che assumerà adesso la seguente forma:

$$Q_1 = \begin{bmatrix} 3 & 3 & 3 \\ 2 & NaN & NaN \\ NaN & NaN & NaN \end{bmatrix}$$

La transizione t_1 ha 3 orologi associati in accordo col grado di abilitazione pari a 3, il cui valore è dovuto al fatto che, essendo stati abilitati a partire dal tempo $\tau = 0$ tutti con valore pari a 5 ed essendo già trascorso un tempo $\tau = 2$, vengono ora aggiornati in relazione al tempo di evoluzione trascorso: da qui si capisce il perchè del loro valore $(5 - 2) = 3$. La transizione t_2, essendo scattata e quindi essendo già stata servita da un servente, risulta essere nuovamente abilitata: le viene associato un nuovo orologio pari a $d_2 = 2$. La transizione che scatta prima è ancora t_2 dopo un tempo $\tau_2 = 2$ e dopo un tempo totale di evoluzione pari a $\tau = 4$; in seguito allo scatto di t_2 dobbiamo aggiornare, come abbiamo già fatto in precedenza, il vettore M, e_{ab} e la matrice Q. La marcatura M_2 raggiunta dopo il tempo evoluzione $\tau = 4$ vale

$$M_2 = [M_2(p_1), M_2(p_2), M_2(p_3)]^T = [1, 0, 2]^T$$

Per questa marcatura, il vettore di abilitazione $e_{ab,2}$ avrà il seguente valore

$$e_{ab,2} = [q(t_1, M_2), q(t_2, M_2), q(t_3, M_2)] = [1, 0, 0]$$

Lo scatto di t_2 ha fatto abbassare ulteriormente il grado di abilitazione di t_1 ma non ha disabilitato la transizione: t_1 ha grado di abilitazione 1, pertanto, avrà

ancora un orologio associato all'interno della matrice Q_2 che assumerà adesso la seguente forma:

$$Q_2 = \begin{bmatrix} 1 \\ NaN \\ NaN \end{bmatrix}$$

Si nota come la sola transizione abilitata a partire dalla marcatura M_2 al tempo $\tau = 4$ sia t_1: quindi t_1 scatta dopo un tempo $\tau_3 = 1$ e dopo un tempo totale di evoluzione pari a $\tau = 5$; in seguito allo scatto di t_1 aggiorniamo il vettore M, e_{ab} e la matrice Q. La marcatura M_3 raggiunta dopo un tempo $\tau_3 = 1$ e tempo totale di evoluzione $\tau = 5$ vale

$$M_3 = [M_3(1), M_3(2), M_3(3)]^T = [0, 1, 2]^T$$

Per questa marcatura, il vettore di abilitazione $e_{ab,3}$ avrà il seguente valore

$$e_{ab,3} = [q(T_1, M_3), q(T_2, M_3), q(T_3, M_3)] = [0, 0, 1]$$

Lo scatto di t_1 ha eliminato tutti i gettoni presenti in p_1 e quindi ha disabilitato la stessa transizione t_1 e la transizione t_2. La sola transizione abilitata a partire dalla marcature M_3 risulta essere t_3 con grado di abilitazione 1, la quale avrà un orologio associato all'interno della matrice

$$Q_3$$

che assumerà adesso la seguente forma:

$$Q_3 = \begin{bmatrix} NaN \\ NaN \\ 3 \end{bmatrix}$$

La transizione t_3 scatta dopo un tempo $\tau_4 = 3$ e dopo un tempo totale di evoluzione pari a $\tau = 8$; in seguito allo scatto di t_3 aggiorniamo il vettore M, e_{ab} e la matrice Q. La marcatura M_4 raggiunta dopo un tempo di clock $\tau_4 = 3$ e tempo di evoluzione totale $\tau = 8$ vale

$$M_4 = [M_4(p_1), M_4(p_2), M_4(p_3)]^T = [6, 0, 1]^T$$

Per questa marcatura, il vettore di abilitazione $e_{ab,4}$ avrà il seguente valore

$$e_{ab,4} = [q(T_1, M_4), q(T_2, M_4), q(T_3, M_4)] = [6, 0, 0]$$

Lo scatto di t_3 elimina 1 gettone sia dal posto p_2 che dal posto p_3 e inserisce 6 gettoni nel posto p_1, riabilitando nuovamente le transizioni t_1 e t_2 e disabilitando

la transizione t_3. La matrice degli orologi Q_4 che assumerà adesso la seguente forma:

$$Q_4 = \begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 \\ 2 & NaN & NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN & NaN & NaN \end{bmatrix}$$

La nostra analisi finisce qua in quanto si voleva valutare l'evoluzione temporale della TPN per un tempo pari a $\tau = 8$.

2.2.3 Conflitto strutturale, effettivo e generale

Definizione 2.2.6. *In una qualsiasi rete di Petri N_d vi è un* conflitto strutturale *se esiste un posto p_i dal quale escono almeno 2 transizioni, cioè se*

$$\exists p_i \in P : \ \|p_i^\bullet\| \geq 2$$

Tutto questo lo indichiamo attraverso la coppia $K = \langle p_i, \{t_x, t_y, \cdots\}\rangle$ dove $t_x, t_y, \cdots \in T$ rappresentano tutte le transizioni che hanno il posto $p_i \in P$ comune in ingresso.

Definizione 2.2.7. *In una qualsiasi rete di Petri N_d vi è un* conflitto effettivo *se esiste un conflitto strutturale K e una marcatura M tale per cui il numero di gettoni nel posto è minore della somma delle ampiezze degli archi in uscita dal posto p_i ed in ingresso a transizioni abilitate,cioè se*

$$\begin{cases} \exists p_i \in P : \ \|p_i^\bullet\| \geq 2 \\ m(p_i) < \sum_{j=1}^{n} Pre(p_i, t_j) \qquad \textit{se } e_{ab}(j) > 0 \end{cases}$$

Un *conflitto effettivo* lo rappresentiamo attraverso la tripla

$$K^E = \langle p_i, \{t_x, t_y, \cdots\}, M\rangle$$

Si noti come la definizione di *conflitto effettivo* valga se e solo se ho al massimo 1 servente per ogni transizione.

Esempio 2.2.3. *Consideriamo l'esempio in Figura 2.13 Analizziamo la TPN rappresentata in (a): il posto p_2 è in* conflitto strutturale *in quanto è collegato sia alla transizione t_1 che alla transizione t_2. Possiamo scrivere cioè che $K = \langle p_2, \{t_1, t_2\}\rangle$; il numero di gettoni presenti in p_2 e il vettore di abilitazione valgono rispettivamente $M(p_2) = 1$ ed $e_{ab} = [1\ 0]$. Andiamo ora a verificare se vi è o no la presenta di* conflitto effettivo*:*

$$Pre(p_2, t_1) = M(p_2) = 1$$

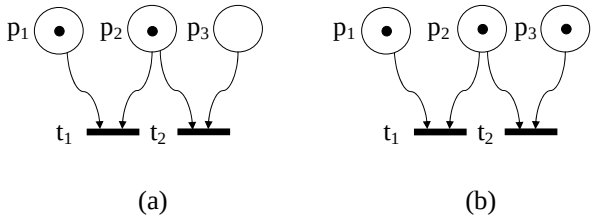


Figura 2.13: (a) Nessun conflitto effettivo. (b) Conflitto effettivo.

Nella rete in Figura (a) *non c'è alcun* conflitto effettivo*. Quindi il numero di gettoni presenti in* p_2 *è sufficiente a far scattare la sola transizione abilitata* t_1*. Nella rete in Figura* (b) *troviamo che il numero di gettoni presenti in* p_2 *e il vettore di abilitazione valgono rispettivamente* $M(p_2) = 1$ *ed* $e_{ab} = [1\ 1]$*; se ora esaminiamo nuovamente la formula che ci permette di stabilire se vi è o no la presenta di* conflitto effettivo *scopriamo che:*

$$Pre(p_2, T_1) + Pre(p_2, T_2) > M(p_2) = 1$$

In questo caso c'è un conflitto effettivo *in quanto il numero di gettoni presenti in* p_2 *è minore della somma degli archi che vanno in ingresso alle transizioni* t_1 *e* t_2 *entrambe abilitate: se scatta* t_1 *non può scattare* t_2 *e viceversa. Risulta chiaro che, se il valore della marcatura nel posto* p_2 *fosse maggiore o uguale a* 2*, non ci sarebbe più la condizione che mi portava ad avere il* conflitto effettivo *e quindi sia la transizione* t_1 *che* t_2 *potrebbero scattare. Quindi, aggiungendo un gettone nel posto* p_2 *abbiamo eliminato il* conflitto effettivo *causato proprio da un numero insufficiente di gettoni.*

Proviamo ora ad analizzare le TPN *in Figura 2.14.*

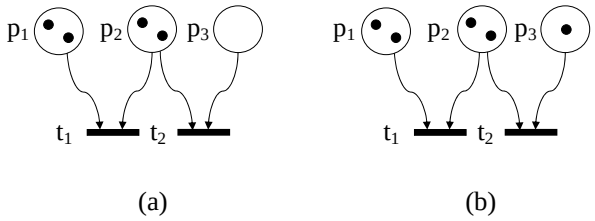


Figura 2.14: (a) Nessun conflitto effettivo. (b) Nessun Conflitto effettivo ma conflitto generale.

Si può notare che il posto p_2 *non è in* conflitto effettivo *in nessuno dei* 2 *casi:*

(a): *Il posto* p_2 *è in conflitto strutturale e la sola transizione abilitata è* t_1*; poichè* $Pre(p_2, t_1) < M(p_2)$ *non ho alcun conflitto effettivo.*

(b): *Il posto p_2 è in conflitto strutturale e le transizioni abilitate sono t_1 e t_2; poichè $Pre(p_2, t_1) + Pre(p_2, t_2) = M(p_2)$ non ho alcun conflitto effettivo.*

Nel caso (b) si presenta, tuttavia, un problema differente: è vero che non vi è alcun conflitto effettivo, ma è anche vero che T_1 è abilitata 2 volte e T_1 è abilitata 1 volta; quindi il numero di gettoni presenti in p_2 non è sufficiente a far scattare tutte le transizioni in relazione al loro grado di abilitazione! Questa situazione è da considerarsi anch'essa come un conflitto che definiremo conflitto generale.

Definizione 2.2.8. *In una qualsiasi rete di Petri temporizzata vi è un* conflitto generale *se esiste un conflitto strutturale K e una marcatura M tale per cui il numero di gettoni nel posto p_i non è sufficiente a far scattare tutte le transizioni in uscita dal posto p_i in accordo con il loro grado di abilitazione,cioè se*

$$\begin{cases} \exists p_i \in P : \ \|p_i^\bullet\| \geq 2 \\ M(p_i) < Pre(p_i, \cdot)e_{ab}^T \end{cases}$$

Un conflitto generale *lo rappresentiamo attraverso la tripla*

$$K^G = \langle p_i, \{t_x, t_y, \cdots\}, M \rangle$$

Il conflitto generale, dunque, è un conflitto che tiene conto del grado di abilitazione delle transizioni ed è quello che noi andremo sempre a verificare in quanto,la sua presenza, genera automaticamente una o più preferenze sullo scatto di alcune transizioni rispetto ad altre; inoltre è quello che più si addice allo studio delle TPN che noi trattiamo in quanto consideriamo reti con limitazione esplicita con una politica di serventi infinita se non è specificato il contrario. Ma cosa succede nel caso in cui possono scattare più transizioni? Consideriamo l'esempio in Figura 2.14 (b): sia t_1 che t_2 possono scattare; questo lo indichiamo come

$$\{t_1 \ t_2\}$$

Siamo in presenza di un conflitto generale *perchè*

$$Pre(p_2, \cdot)e^T = \ [2] \ \begin{bmatrix} 2 \\ 1 \end{bmatrix} \ = 3 > M(p_2).$$

Quindi in presenza di un conflitto generale *dobbiamo fare delle scelte: dobbiamo decidere quali transizioni devono scattare e quante volte farle scattare in quanto il numero di gettoni nei posti non è sufficiente per lo scatto di tutte; tuttavia continuiamo ad applicare l'algoritmo 2.2.1 applicando tale scelta al passo 8) (come spiegheremo nel paragrafo successivo), assumendo che nonostante le transizioni possano essere servite contemporaneamente ($[t_1 \ t_2]$), scattino sequenzialmente ($[t_1 \ t_2]$ oppure $[t_2 \ t_1]$); Quindi*

$$\{t_1 \ t_2\} = \{ \ [t_1 \ t_2] \text{ oppure } [t_2 \ t_1]$$

2.2.4 Conflitto generale: possibili risoluzioni

Finora abbiamo detto che nelle TPN si possono incontrare dei conflitti generali ma non abbiamo detto come possiamo risolverli; ci sono 2 tecniche:

1. **Stocastica.**
2. **Deterministica.**

Risoluzione stocastica: Applichiamo il seguente algoritmo

Algoritmo 2.2.2. *Questo algoritmo si basa sull'assegnazione di un peso alle transizioni della rete:*

1. *Applicare l'algoritmo 2.2.1 sostituendo il passo* $8)$ *con i seguenti passi.*
2. *Inizilizzare* $x = 1$*;*
3. *Individuare* o^**, mediante la relazione*

$$o^* = \min_{j:t_j \in T} \{o_i\}$$

come il valore più piccolo fra gli orologi o_j *associati alle transizioni* t_j *abilitate nella marcatura* M_l*.*

4. *Memorizzare le transizioni corrispondenti a* o^* *all'interno del vettore di scatto denominato* se_fire_x*.*
5. *Memorizzare la marcatura corrente* $M_{l,x} = M_l$ *e il vettore dei serventi* $s_x = s$*.*
6. *Memorizzare il peso associato a ciascuna transizione all'interno di un vettore* α *di dimensione* $1 \times n_d$
7. *Calcolare la probabilità di scatto di ogni transizione* t_j *abilitata, dipendente sia dal peso* $\alpha(j)$ *che dall'effettivo grado di abilitazione* $se\_fire_x(j)$*, come*

$$Prob\_T_d = \frac{\alpha. * se\_fire_x^T}{\sum(\alpha. * se\_fire_x^T)}.$$

8. *Per ogni transizione abilitata* $t_j \in T$ *si estrae un valore random da una distribuzione uniforme compresa fra* $[0, Prob\_T_d(j)]$ *e lo si memorizza in un nuovo vettore* $\vec{con}$ *di dimensioni* $1 \times n_d$ *dato da*

$$con(j) = unifrnd(0, Prob\_T_d(j)) \qquad t_j \in T.$$

9. *Calcolare il minimo valore* con^* *del vettore* $\vec{con}$ *e l'indice ad esso associato che mi indicherà la transizione con priorità maggiore*

$$con^* = \max_{j:t_j \in T} con(j) \qquad t_j \in T$$

10. *Creare un vettore nullo* se_single_fire *di dimensione* $1 \times nd$ *che avrà un valore pari a* 1 *in corrispondenza della colonna* j*-esima se* $con(j) = con^*$*; se più transizioni hanno* $con(j) = con^*$*, allora si sceglie quella con indice più basso.*
11. *Aggiornamento del vettore della sequenza di scatto*

$$\vec{\sigma} = \vec{\sigma} + se\_single\_fire$$

.

12. *Aggiornare il vettore della marcatura attuale* $M_{l,x+1} = M_{l,x} + Pre \cdot se\_single\_fire$ *in seguito alla scatto della transizione con priorità maggiore.*
13. *Incrementare l'indice* $x = x + 1$.
14. *Aggiornare i serventi che sono già stati utilizzati*

$$s_x = s_{x-1} - se\_single\_fire.$$

15. *Calcolare il grado di abilitazione di ogni transizione abilitata in funzione del numero di serventi associati alle transizioni stesse*

$$se\_fire_x(j) = \min(s_x(j), se\_fire_x(j)) \quad t_j \in T$$

Se $\exists se\_fire_x(j) > 0 \quad t_j \in T$ *allora vai al passo* 7*), altrimenti vai al passo successivo.*
16. *Proseguire l'algoritmo 2.2.1 dal passo* 9).

Precisazioni:Il fatto che una transizione abbia un peso maggiore non vuol dire che scatti prima di un altra con peso minore; bisogna considerare che la probabilità di scatto associata ad ogni transizione è in funzione sia del suo peso che del grado di abilitazione che essa assume (punto 7 dell'algoritmo). Tuttavia, nonostante una transizione abbia una probabilità di scatto di scatto maggiore rispetto a tutte le altre, non è detto che scatti prima: infatti la scelta dello scatto è dovuta ad una funzione random uniforme (da qui deriva il nome di *metodo stocastico*) che prende dei valori casuali (punto 8 dell'algoritmo) di cui poi vado a considerare il minimo (punto 9 dell'algoritmo). Si noti che se il vettore $\vec{con}$ dovesse avere al suo interno più valori uguali pari al $con^* = \min(con)$ (ipotesi remota ma possibile) vado a scegliere il valore memorizzato con l'indice più basso.

Risoluzione deterministica: Applichiamo il seguente algoritmo

Algoritmo 2.2.3. *Questo algoritmo si basa sull'assegnazione sia di un peso che di un livello di priorità alle transizioni della rete. Definisco la funzione*

$$\Pi : T \rightarrow \mathbb{N}_+$$

dove $\Pi(t_j)$ *è la priorità della transizione* t_j*; se* $\Pi(t_j) < \Pi(t_k)$ *allora* t_j *ha priorità rispetto a* t_k*.*

1. *Applicare l'algoritmo 2.2.1 sostituendo il passo* $8)$ *con i seguenti passi.*
2. *Inizilizzare* $x = 1$*;*
3. *Individuare* o^**, mediante la relazione*

$$o^* = \min_{j:t_j \in T} \{o_i\}$$

   come il valore più piccolo fra gli orologi o_j *associati alle transizioni* t_j *abilitate nella marcatura* M_l*.*
4. *Memorizzare le transizioni corrispondenti a* o^* *all'interno del vettore di scatto denominato* se_fire_x*.*
5. *Memorizzare la marcatura corrente* $M_{l,x} = M_l$ *e il vettore sei serventi* $s_x = s$*.*
6. *Memorizzare il peso associato a ciascuna transizione all'interno della prima riga della matrice* α *e il livello di priorità a cui appartengono le transizioni all'interno della seconda riga della matrice* α*, dove* $\alpha(2, j) = \Pi(t_j)$*; quindi* α *avrà una dimensione pari a* $2 \times n_d$
7. *Calcolare l'insieme delle transizioni abilitate che appartengono al livello di priorità più basso.*

$$priority\_level = \min(se\_ones^{(T)}. * \alpha(2, \cdot)$$

   dove se_ones *è un vettore colonna di dimensione* $1 \times n_d$ *che memorizza un valore pari a* 1 *in corrispondenza della riga* j*-esima se la transizione* t_j *è abilitata altrimenti memorizza* 0*.*
8. *Se solo una transizione* t_j *verifica la condizione* $alpha(2, t_j) = priority\_level$ *allora si deve creare un vettore nullo* se_single_fire *di dimensione* $1 \times n_d$ *che avrà un valore pari a* 1 *in corrispondenza della colonna* j*-esima se* $alpha(2, t_j) = priority\_level$ *e si deve andare al passo* 12*); altrimenti vai al passo successivo.*

9. *Calcolare la probabilità di scatto di ogni transizione t_j abilitata tale per cui $\alpha(2, t_j) = priority\_level$, dipendente sia dal peso $\alpha(j)$ che dall'effettivo grado di abilitazione $se\_fire_x(j)$, come*

$$Prob\_T_d = \frac{\alpha. * se\_fire_x^T}{\sum(\alpha. * se\_fire_x^T)}.$$

10. *Per ogni transizione abilitata $t_j \in T$ per la quale vale $\alpha(2, t_j) = priority\_level$, si estrae un valore random da una distribuzione uniforme compresa fra $[0, Prob\_T_d(j)]$ e lo si memorizza in un nuovo vettore $\vec{con}$ di dimensioni $1 \times nd$ dato da*

$$con(j) = unifrnd(0, Prob(j)) \quad t_j \in T.$$

11. *Calcolare il minimo valore con^* del vettore $\vec{con}$ e l'indice ad esso associato che mi indicherà la transizione con priorità maggiore*

$$con^* = \min_{j:t_j \in T} con(j) \quad t_j \in T$$

12. *; se più transizioni hanno $con(j) = con^*$, allora si sceglie quella con indice più basso.*
13. *Aggiornamento del vettore della sequenza di scatto*

$$\vec{\sigma} = \vec{\sigma} + se\_single\_fire$$

.

14. *Aggiornare il vettore della marcatura attuale $M_{l,x+1} = M_{l,x} + Pre \cdot se\_single\_fire$ in seguito alla scatto della transizione con priorità maggiore.*
15. *Incrementare l'indice $x = x + 1$.*
16. *Aggiornare i serventi che sono già stati utilizzati*

$$s_x = s_{x-1} - se\_single\_fire.$$

17. *Calcolare il grado di abilitazione di ogni transizione abilitata in funzione del numero di serventi associati alle transizioni stesse*

$$se\_fire_x(j) = \min_{s_x(j) \neq 0} (s_x(j), se\_fire_x(j)) \quad t_j \in T$$

Se $\exists se\_fire_x(j) > 0 \quad t_j \in T$ allora vai al passo 7), altrimenti vai al passo successivo.

18. Proseguire l'algoritmo 2.2.1 dal passo 9).

Precisazioni:E' possibile risolvere un eventuale conflitto generale grazie all' appartenenza ad un certo livello di priorità; le transizioni che appartengono ad un livello di priorità minore scattano prima di quelle che appartengono ad un livello di priorità maggiore. Tuttavia, se 2 transizioni che devono scattare, sono in conflitto generale e appartengono allo stesso livello di priorità, è necessario introdurre come secondo elemento risolutivo per la scelta dello scatto di una delle 2 transizioni: il peso associato alle stesse transizioni. Stiamo, cioè, introducendo un metodo di risoluzione stocastico. Per noi, nonostante possa verificarsi quest'ultimo caso, rimane lo stesso un metodo di risoluzione deterministico per via dei livelli di priorità. Chiaramente l'esempio appena fatto soltanto considerando 2 transizioni, può essere esteso a un numero maggiore. E' facile capire che il caso di risoluzione stocastica è un caso particolare di quello deterministico in quanto stiamo considerando le transizioni appartenenti tutte a un unico livello di priorità.

Consideriamo un esempio per chiarire meglio l'applicazione dei 2 algoritmi

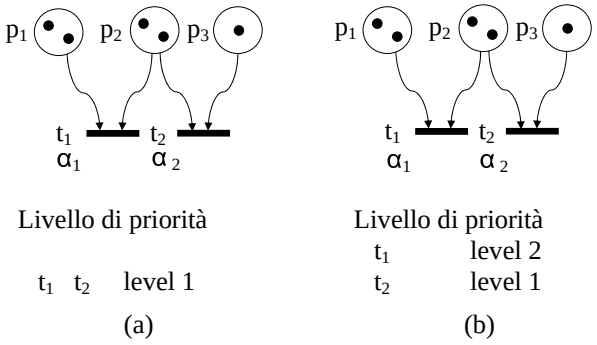


Figura 2.15: a) Conflitto generale: risoluzione con il metodo stocastico. b) Conflitto generale: risoluzione con il metodo deterministico.

Esempio 2.2.4. *Per entrambe le reti in figura si hanno le seguenti proprietà:*

1. $M_0 = [2, 2, 1]$
2. $e_{ab,0} = [2, 1]$
3. $s_0 = [0, 0]$

Consideriamo il caso in Figura 2.15 a): Le transizioni abilitate t_1 e t_2 appartengono tutte allo stesso livello di priorità, quindi non serve specificare la seconda riga della matrice α in quanto risulterebbe una informazione inutile ai fine della risoluzione del conflitto. Il vettore α avrà la seguente forma:

$$\alpha = [\alpha_1, \alpha_2].$$

*Se ipotizziamo che $\alpha_1 > \alpha_2$ ci sono buone possibilità che scatti 2 volte la transizione t_1, mentre, se ipotizziamo $\alpha_1 < \alpha_2$, ci sono buone possibilità che scatti prima la transizione t_2 e poi la transizione t_1 (1 sola volta poichè il grado di abilitazione di t_1 si è abbassato in seguito allo scatto di t_2). Tutta la dinamica di scatto è legata al vettore **con** che abbiamo introdotto nel punto 9) e 10) dell'algoritmo 2.2.2: si può notare proprio da questo aspetto che la tecnica di risoluzione risulta essere stocastica. Consideriamo il caso in Figura 2.15 b): Le transizioni abilitate t_1 e t_2 appartengono a 2 livelli di priorità: α sarà una matrice che avrà la prima riga costituita dai pesi delle transizioni e la seconda riga costituita dai livelli di priorità associati alle transizioni stesse, cioè*

$$\alpha = \begin{bmatrix} \alpha_1 & \alpha_2 \\ 1 & 2 \end{bmatrix}.$$

Scattano prima le transizioni abilitate con livello di priorità più basso. Nell'esempio in Figura 2.15 b) scatta prima la transizione t_2 (1 volta) e poi la transizione t_1 (1 volta) sia che ipotizzaimo $\alpha_1 > \alpha_2$e sia che ipotizziamo $\alpha_1 < \alpha_2$. In questo caso conosco a priori le transizioni che scattano: questo grazie ai livelli di priorità diversi associati alle transizioni.

2.2.5 Reti di Petri temporizzate stocastiche

Le reti di Petri temporizzate stocastiche sono reti di Petri in cui i tempi di scatto delle transizioni sono rappresentati da variabili aleatorie. Una TPN stacastica si può pertanto considerare come un modello che genera un processo stocastico: a ogni transizione temporizzata è associato un ritardo descritto da una variabile aleatoria che ha una certa distribuzione, quindi associamo a ogni transizione t_j il parametro λ_j caratteristico della sua distribuzione, detto *frequenza* o *tasso di scatto* che corrisponde all'inverso del ritardo medio di scatto (si consulti 6.1.2).

Definizione 2.2.9. *Una rete di Petri temporizzata stocastica è una struttura algebrica $R = (N_d, Tempo)$ dove:*

- *$N = (Pre, Post)$ è una rete posto/transizione come nella definizione 2.1.1*

- *$Tempo$: $T \longrightarrow \mathbb{R}_0^+$ è una funzione associata al set di transizioni T secondo la quale $Tempo(t_j) = \frac{1}{\lambda_j} = \mu_j$ = valor medio associato a t_j; definisco $\lambda = [\lambda_1\ \lambda_2\ \cdots]$ come il vettore delle frequenze di scatto delle transizioni.*

Per una TPN le regole di scatto sono le stesse di una TPN, con l'unica differenza che, in questo caso, la scelta della prossima transizione da far scattare viene effettuata sulla base delle probabilità di scatto delle singole transizioni. La matrice degli orologi Q avrà dei valori, associati alle transizioni temporizzate stocastiche, che vengono estrappolati di volta in volta in modo casuale dalle rispettive distribuzioni. La gestione degli orologi è come per le TPN è deterministiche: il comportamento delle TPN stocastiche evolve come se tutte le volte che essa raggiunge una marcatura nuova, ogni transizione t_j abilitata ricampionasse dalla propria densità di probabilità del ritardo una nuova istanza d_j. Ad esempio, se analizziamo le transizioni temporizzate aventi un ritardo descritto da una variabile aleatoria con funzione distribuzione esponenziale negativa, a ogni transizione t_j dobbiamo associare il parametro $\lambda_j = \frac{1}{\mu_j}$: se volessi attribuire un orologio all'interno della matrice Q, dovrei andare a estrapolare in modo random un valore dalla distribuzione esponenziale negativa avente valor medio μ. Molti ricercatori hanno dimostrato le potenzialità delle TPN stocastiche con distribuzione esponenziale per l'analisi delle prestazioni di sistemi reali, risultanti soprattutto dal fatto che, dal punto di vista del comportamento dinamico, una TPN stocastica con distribuzione esponenziale è equivalente ad una catena di Markov a tempo continuo: i tempi di permanenza in ogni marcatura sono distribuiti in modo esponenziale. Per studiare analiticamente le reti di Petri discrete stocastiche esponenziali siamo costretti a ricondurci alla una catena di Markov a tempo continuo equivalente. E' facile capire, quindi, come il problema maggiore nella valutazione di indici di prestazioni utilizzando modelli TPN stocastici con transizioni a ritardo esponenziale sta nella necessità di lavorare con le equazioni di equilibrio basate sul grafo di raggiungibilità. Infatti, le dimensioni del grafo di raggiungibilità crescono in modo esponenziale sia con il numero di marche nella marcatura iniziale M_0 sia con il numero dei posti: pertanto le dimensioni di tale grafo e l'onere computazionale della procedura di soluzione impediscono di ottenere una soluzione analitica esatta. Tuttavia limitare lo studio dei sistemi reali attraverso le TPN con transizioni stocastiche esponenziali risulta abbastanza riduttivo! Se si vuole studiare un modello attraverso reti di Petri temporizzate con transizioni stocastiche che non abbiano andamento esponenziale, si è costretti a simularle in quanto analiticamente lo studio risulterebbe troppo oneroso e complesso.

L'evoluzione nel tempo di una TPN stocastica con distribuzione esponenziale ci permette, inoltre, di studiare i sistemi a coda, dove ogni coda è costituita da:

- m_d servitori (o serventi) identici;

- clienti dello stesso tipo;
- uno spazio in cui i clienti attendono il servizio, che è poi la coda vera e propria.

Lo studio di questi sistemi sarà facilitato dall'utilizzo del simulatore $HYPENS$ grazie al quale possiamo avere diverse statistiche simulando direttamente la rete di Petri stocastica e non essendo costretti a calcolarcele analiticamente attraverso l'utilizzo delle tabelle: si consulti il paragrafo **??**.

Capitolo 3

Reti di Petri continue

3.1 Introduzione alle reti di Petri continue

Come abbiamo spiegato nel capiloto, il numero di stati che una rete di Petri può raggiungere cresce indefinitamente con il numero di marche nella rete e ciò costituisce una limitazione importante all'uso pratico dei modelli di SED costituiti da reti di Petri. Per fare un esempio, si consideri un sistema di produzione costituito da 3 macchine in serie m_1, m_2, m_3 intervallate da 2 spazi di accodamento intermedi di capacità c_1 e c_2: tutti i pezzi visitano le macchine in sequenza e attendono in coda se necessario. Nell'ipotesi che ci sia spazio di accodamento illimitato a monte di m_1 e a valle di m_3, il sistema può raggiungere un numero di stati pari a $2^3(c_1 + 1)(c_2 + 1)$; se per esempio $c_1 = c_2 = 10$, gli stati in cui il semplice sistema produttivo può trovarsi sono ben 968, il che rende bene l'idea di quale possa diventare questo numero per sistemi appena più complicati. E' anche per ovviare a questo inconveniente che sono stati introdotte le reti di Petri continue (CPN), in cui le marcature sono numeri reali e il processo di scatto delle transizioni è continuo. Le CPN costituiscono un'approssimazione accettabile del comportamento delle parti di un SED che possono essere assimilati a processi continui; per esempio per tornare alla catena di produzione precedentemente descritta, se la produzione avviene senza interruzioni o quasi, è ragionevole approssimare come continuo il flusso delle parti da lavorare attraverso la serie delle 3 macchine, così come è di conseguenza plausibile considerare reale il numero di pezzi che attendono nelle diverse code. Sarebbe invece sbagliato comportarsi in maniera analoga per quanto riguarda lo stato operativo di ogni macchina (attiva, guasta, in riparazione, $\cdots$) che deve forzatamente essere rappresentato da una variabile discreta. Perchè, quindi, introdurre le reti di Petri continue? Perchè i problemi di scala reale sono spesso intrattabili sia analiticamente che computazionalmente se si usano modelli discreti: per far fronte a questo problema, si sono approssimati i

modelli discreti con modelli continui che hanno portato i seguenti vantaggi:

1. Aumento dell'efficienza computazionale in quanto la simulazione di modelli fluidi è più efficiente;
2. Riduzione della dimensione dello spazio di stato;
3. Introduzione, nei modelli, dei parametri continui e relativo aumento della velocità di ottimizzazione;
4. L'approssimazione di fluido non introduce errori significativi.

3.2 Reti di Petri continue

Una rete di Petri continua temporizzata può essere considerata, idealmente, come un limite per $k \to \infty$ di una TPN nella quale ogni marca è stata suddivisa in k parti infinitesime ciascuna detta *token*; questo procedimento consente di trasformare le marche discrete in quantità reali (token). Conseguentemente, in una rete di Petri continua sia la marcatura sia i pesi degli archi sono costituiti da numeri reali. Consideriamo l'esempio in Figura 3.1.

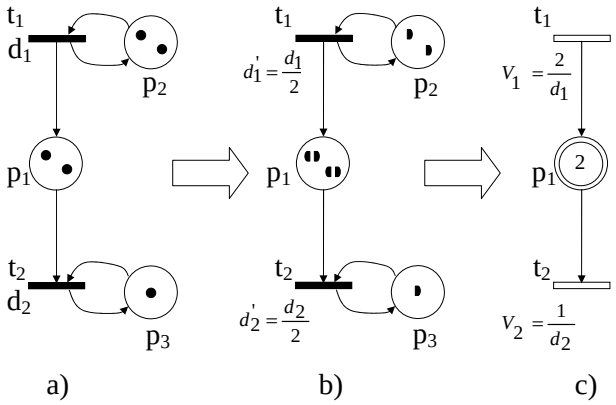


Figura 3.1: Da una rete di Petri temporizzata ad una rete di Petri continua.

Esempio 3.2.1. *In Figura 3.1 a) Si ha un flusso di marche (o gettoni) attraverso il posto p_1 e le transizioni t_1 e t_2 sono rispettivamente a doppio servente (posto p_2) e a singolo servente (posto p_3): ogni volta che è trascorso un tempo $\tau = d_1$ vengono depositate 2 marche in p_1 mentre ogni volta che è trascorso un tempo $\tau = d_2$ viene rimossa 1 marca da p_1. In altre parole, il tasso di flusso delle marche attraverso le transizioni vale:*

- $\frac{2}{d_1}$ *per la transizione* t_1*;*
- $\frac{1}{d_2}$ *per la transizione* t_2 *finchè ci sono marche in* p_1*, altrimenti vale* 0*.*

La Figura 3.1 b*) è ottenuta dalla Figura 3.1* a*) nel seguente modo: ogni marcatura è divisa in* $k = 2$ *tokens e il tempo associato a ogni transizione è a sua volta diviso in* 2*. Ogni volta che è trascorso un tempo* $\tau = \frac{d_1}{2}$ *vengono depositati* 2 *tokens in* p_1 *mentre ogni volta che è trascorso un tempo* $\tau = \frac{d_2}{2}$ *viene rimosso* 1 *token da* p_1*. In altre parole, il tasso di flusso dei token attraverso le transizioni vale:*

- $\frac{4}{d_1}$ *per la transizione* t_1*;*
- $\frac{2}{d_2}$ *per la transizione* t_2 *finchè ci sono token in* p_1*, altrimenti vale* 0*.*

Il tempo di riempimento e svuotamento del posto p_1 *è analogo sia per la Figura 3.1* a*) che* b*). Se ogni marca in* p_1 *è divisa in* k *tokens, il tempo associato alla transizione* t_1 *e* t_2 *è rispettivamente* $\tau = \frac{d_1}{k}$ *e* $\tau = \frac{d_2}{k}$*. Quando* k *tende ad infinito, l'unità marca può essere usata al posto dell'unità token: la marcatura di* p_1 *è espressa in marche e il rate del flusso di tokens* $\frac{k}{d_1}$ *e* $\frac{k}{d_2}$ *è rimpiazzato attraverso il rate del flusso di marche* $V_1 = \frac{2}{d_1}$ *e* $V_2 = \frac{1}{d_2}$*, dove* V_j *indica la* massima velocità *di scatto associata alla transizione* t_j *come mostrato in Figura 3.1* c*). Se le transizioni nella rete sono più di una, posso definire un vettore dell velocità massime* V *la cui generica componente* V_j *memorizza il valore della velocità massima associata alla transizione* j*-esima.*

Definizione 3.2.1. *La velocità massima associata ad una transizione è data dal prodotto fra il rate di flusso associato al singolo servente per il numero di serventi associati alla transizione stessa.*

Consideriamo, ad esempio, in Figura 3.1 c) la transizione t_2 con velocità massima $V_2 = \frac{1}{d_2}$: se $M(p_1) = 0$ non possiamo parlare più di velocità massima, perchè, in quest'ultimo caso, la velocità sarebbe nulla. Introduciamo allora il concetto di vettore delle *velocità istantanea di scatto* $\vec{v}$ (detto anche IFS, *istantaneous firing speed*) dove $v_j = \vec{v}(t_j)$ indica il valore attuale della velocità istantanea associata alla transizione j-esima: se $M(p_1) = 0$ allora $v_1 = 0$ e se $M(p_1) > 0$ allora $v_1 = V_1$.

Definizione 3.2.2. *Una* PN *continua è una struttura* $N_c = (P, T, Pre, Post)$ *dove:*

- $P = \{p_1, p_2, \cdots, p_{mc}\}$ *è un insieme finito di* m_c posti *continui non vuoto.*
- $T = \{t_1, t_2, \cdots, t_{nc}\}$ *è un insieme finito di* n_c transizioni *continue non vuoto.*

- $Pre : P \times T \longrightarrow \mathbb{R}_0^+$*: è la* funzione di pre-incidenza *che specifica gli archi diretti dai posti alle transizioni (detti archi* Pre*) e viene rappresentata mediante una matrice* $m_c \times n_c$*; più precisamente,* $Pre(p_i, t_j)$ *indica quanti archi vanno dal posto* p_i *alla transizione* t_j*.*

- $Post : P \times T \longrightarrow \mathbb{R}_0^+$*: è* funzione di post-incidenza *che specifica gli archi diretti dalle transizioni ai posti (detti archi* $Post$*) e viene rappresentata mediante una matrice* $m_c \times n_c$*; più precisamente,* $Post(p_i, t_j)$ *indica quanti archi vanno dalla transizione* t_j *al posto* p_i *.*

Si noti che in una rete di Petri continua $P_c = P$ e $T_c = T$, cioè l'insieme di tutti i posti e transizioni della rete sono continui. Le matrici Pre e $Post$ sono delle matrici di numeri reali non negativi. Si denota con $Pre(\cdot, t_j)$ la colonna della matrice Pre relativa alla transizione t_j e con $Pre(p_i, \cdot)$ la riga della matrice Pre relativa al posto p_i. La stessa notazione vale per la matrice $Post$. Si suppone che $P \cap T = \emptyset$, cioè posti e transizioni sono insiemi disgiunti e che $P \cup T \neq \emptyset$, cioè la rete è costituita da almeno un posto o da una transizione. L'informazione sulla struttura di rete contenuta nelle matrici Pre e $Post$ può essere compattata in un'unica matrice, detta di incidenza.

Definizione 3.2.3. *Una* PN *continua temporizzata è caratterizzata dalla struttura algebrica* $N_{cv} = (N_c, V)$ *dove:*

- $N_c = (P, T, Pre, Post)$ *definita nella definizione ;*

- $V : T \longrightarrow \mathbb{R}_0^+$ *è una funzione che, data una transizione* j*-esima mi restituisce la velocità massima associata,* $V(t_j) = V_j$*.*

Definizione 3.2.4. *Data una rete* N_{cv}*, con* m_c *posti ed* n_c *transizioni, la* matrice di incidenza $C : P \times T \longrightarrow \mathbb{R}_0^+$ *è la matrice* $m_c \times n_c$ *definita come:*

$$C = Post - Pre$$

cioè il generico elemento di C *vale* $C(p_i, t_j) = Post(p_i, t_j) - Pre(p_i, t_j)$*.*

Data C non posso ricostruire il grafo, mentre date le matrici Pre e $Post$ posso ricostruire perfettamente il grafo. Mediante la marcatura è possibile definire lo stato di una CPN.

Definizione 3.2.5. *Una* marcatura *è una funzione* $M : P \longrightarrow \mathbb{R}_0^+$ *che assegna ad ogni posto un volume di fluido.*

Definizione 3.2.6. *Una rete* N_{cv} *con una marcatura iniziale* M_0 *è detta* rete marcata o sistema di rete *e viene indicata come* $R_c = \langle N_{cv}, M_0 \rangle$ *.*

Si noti che il vettore della generica marcatura M è un vettore colonna $m_c \times 1$, quindi ha tante righe quanti sono i posti continui della rete. Un esempio chiarirà questi concetti.

Esempio 3.2.2. *In Figura 3.2 è rappresentata la rete $N_{cv} = (P, T, Pre, Post, V, M_0)$ con insieme dei posti $P = \{p_1\}$ e insieme delle transizioni $T = \{t_1, t_2\}$.*

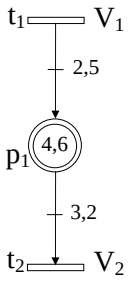


Figura 3.2: Una rete di Petri continua temporizzata marcata.

Le matrici Pre e $Post$ valgono:

$$Pre = \begin{bmatrix} 0 & 3,2 \end{bmatrix}, \qquad Post = \begin{bmatrix} 2,5 & 0 \end{bmatrix}.$$

La matrice di incidenza vale:

$$C = \begin{bmatrix} 2,5 & -3,2 \end{bmatrix}.$$

Il vettore delle velocità massime V e della marcatura iniziale M_0 vale rispettivamente $V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$ $M_0 = [4, 6]$

Anche nel caso di CPN possiamo definire le seguenti simbologie. Data una transizione si definiscono i seguenti insiemi di posti:

- Insieme dei posti in ingresso alla transizione t_j

$$^\bullet t_j = \{p_i \in P \mid Pre(p_i, t_j) > 0\}.$$

- Insieme dei posti in uscita dalla transizione t_j

$$t_j^\bullet = \{p_i \in P \mid Post(p_i, t_j) > 0\}.$$

Dato un posto si definiscono i seguenti insiemi di transizioni:

- Insieme delle transizioni in ingresso al posto p_i

$$^\bullet p_i = \{t_j \in T \mid Post(p_i, t_j) > 0\}.$$

- Insieme delle transizioni in uscita dal posto p_i

$$p_i^\bullet = \{t_j \in T \mid Pre(p_i, t_j) > 0\}.$$

Ad esempio nella rete in Figura 3.1 vale ${}^\bullet t_2 = \{p_1\}$, $t_1^\bullet = \{p_1\}$, ${}^\bullet p_1 = \{t_1\}$, $p_1^\bullet = \{t_2\}$.

Definizione 3.2.7. *Sia M una marcatura di una data CPN; l'insieme di posti P può essere suddiviso in 2 sott'insieme:*

- *$P^+(M)$ è l'insieme di posti p_i tali per cui $M(p_i) > 0$;*
- *$P^0(M)$ è l'insieme di posti p_i tali per cui $M(p_i) = 0$.*

*Una **macro-marcatura** è l'unione di tutte le marcature M con lo stesso set $P^+(M)$ di posti marcati: il numero di macro-marcature raggiungibile attraverso una CPN con nc posti è minore o uguale a 2^{nc}.*

Una macro-marcatura può essere indicata come M^*. Il cambio da una macro-marcatura ad un'altra si verifica quando si manifesta uno dei seguenti eventi:

- **C1-event**: la marcatura di un posto marcato diventa nulla.
- **C2-event**: un posto non marcato diventa marcato.

Consideriamo l'esempio in Figura 3.2: le possibili macro-marcature sono 2, cioè $P^+(M_0^*) = \{p_1\}$ e $P^+(M_1^*) = \emptyset$.

3.2.1 Abilitazione ed evoluzione

Definizione 3.2.8. *Data una rete $\langle N_c, M\rangle$, una* transizione t_j *si dice*

- ***fortemente abilitata** al tempo τ se*

$$M(p_i) > 0 \quad \forall p_i \in^\bullet t_j$$

- ***debolmente abilitata** al tempo τ se*

$$M(p_i) = 0 \quad \forall p_i \in^\bullet t_j$$

Quindi se ogni posto $p_i \in P$ in ingresso a t_j contiene un numero di marche maggiore di zero allora la transizione t_j è fortemente abilitata, altrimenti è debolmente abilitata. Consideriamo l'esempio in Figura 3.2: sia t_1 che t_2 sono fortemente abilitate.

Definizione 3.2.9. *Data una rete* $\langle N_c, M \rangle$*, definiamo la* velocità *come:*

- ***velocità di riempimento*** *del posto* p_i

$$I_i = \sum_{t_j \in {}^\bullet p_i} Post(p_i, t_j) \cdot v_j$$

- ***velocità di svuotamento*** *del posto* p_i

$$O_i = \sum_{t_k \in {}^\bullet p_i} Post(p_i, t_k) \cdot v_k$$

Definizione 3.2.10. *Per calcolare la velocità associata a ogni transizione al tempo* τ *seguiamo i seguenti passi:*

- *Se una transizione* t_j *è fortemente abilitata, può scattare alla velocità massima:* $v_j(\tau) = V_j$*.*
- *Sia* t_j *una transizione debolmente abilitata e sia* $A_j(\tau)$ *il sotto insieme dei posti* $p_i \in^\bullet t_j$ *tali per cui* $M(p_i) = 0$*. Se* t_j *non è una transizione in conflitto generale rispetto ai posti in* $A_j(\tau)$*, allora la velocità istantanea di scatto di* t_j *vale*

$$v_j(\tau) = \min_{p_i \in A_j(\tau)} \left(\frac{1}{Pre(p_i, t_j)} \sum_{t_k \in {}^\bullet p_i} Post(p_i, t_k) \cdot v_k(\tau), \;\; V_j \right)$$

Questo sta a giustificare il fatto che, per tutti i posti vuoti, la velocità totale di svuotamento non può mai essere maggiore della velocita totale di riempimento. Se una transizione abilitata debolmente è coinvolta in un conflitto effettivo, la sua velocità istantanea di scatto può essere minore della quantità indicata nella definizione 3.2.10. Consideriamo la rete in Figura 3.3 Per un tempo $\tau > 0$ sia la transizione t_1 che la transizione t_2 sono fortemente abilitate, mentre la transizione t_3 è debolmente abilitata (non al tempo $\tau = 0$ ma subito dopo): le velocità istantanee di scatto per $\tau > 0$ valgono:

1. $v_1 = V_1 = 1$;
2. $v_2 = V_2 = 3$;
3. $v_3 = \min(v_1, v_2, V_3) = v_1 = 1$.

Fino ad adesso non abbiamo ancora detto nulla su come evolve una rete continua temporizzata: introduciamo il concetto di bilanciamento assumendo che $M(p_i) = M_i(\tau)$ per mettere in evidenza il fatto che si tratta di marcature che variano con continuità nel tempo.

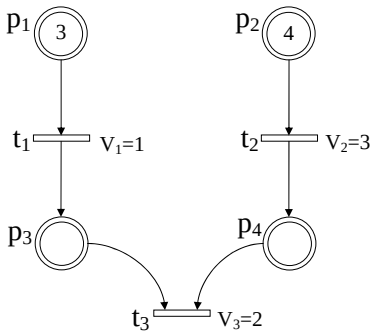


Figura 3.3: Esempio di CPN con transizioni sia fortemente abilitate che debolmente abilitate.

Definizione 3.2.11. *Il **bilanciamento** della marcatura del posto p_i in una CPN è:*

$$B_i = I_i - O_i = C(i, \cdot)\vec{v}$$

cioè la differenza fra le velocità di riempimento e di svuotamento di un posto. Il bilanciamento di p_i corrisponde alla derivata temporale della sua marcatura:

$$\frac{dM_i(\tau)}{dt} = B_i(\tau) = \frac{M_i(\tau + dt) - M_i(\tau)}{dt}$$

*L'**evoluzione** della marcatura nel tempo è descritta come*

$$M_i(\tau + dt) = M_i(\tau) + B_i(\tau) \cdot dt$$

Per ogni posto p_i vuoto deve valere $B_i \geq 0$, cioè la velocità totale di svuotamento deve essere minore o al massimo uguale alla velocità totale di riempimento.

Consideriamo l'esempio in Figura 3.3 a partire dall'istante di tempo $\tau_0 = 0$:

$$\begin{array}{lll} M_1(\tau_0 + t) = 3 - t & \text{per} & t \in [0,\ 3] \\ M_2(\tau_0 + t) = 4 - 3t & \text{per} & t \in [0,\ \frac{4}{3}] \\ M_3(\tau_0 + t) = 0 & \text{per} & t \in [0,\ \frac{4}{3}] \\ M_4(\tau_0 + t) = 2t & \text{per} & t \in [0,\ \frac{4}{3}] \end{array}$$

Queste sono alcune delle possibili equazioni che regolano il flusso all'interno della rete: per studiare l'evoluzione delle CPN si deve introdurre il concetto di **IB-state** (*invariant behavior state*).

Definizione 3.2.12. *Si definisce IB-state lo stato delle rete di Petri continue all'interno del quale il vettore delle velocità istantanee $\vec{v}$ rimane costante. L'evoluzione*

consiste nel cambio nel cambio da uno stato di velocità ad un altro stato di velocità attraverso il verificarsi di un evento che cambia il valore della marcatura alla quale è associato. Se si possono verificare più eventi, facciamo evolvere la rete in funzione dell'evento che si verifica prima di tutti.

Si noti come il passaggio da un IB-state ad una altro è dovuto al verificarsi dell'evento **C1**, cioè al fatto che la marcatura di un posto marcato diventa nulla. Consideriamo, nuovamente, l'esempio in Figura 3.3 a partire dall'istante di tempo $\tau_0 = 0$:

1. IB-state 1 al tempo $\tau_0 = 0$ costituito dalla marcatura $M = (3,\ 4,\ 0,\ 0)$ e da un vettore delle velocità istantanee $\vec{v} = (1,\ 3,\ 3)$. Le equazioni che regolano l'evoluzione a partire dal tempo $\tau_0 = 0$ sono:

$$\begin{aligned} M_1(\tau_0 + t) &= 3 - t \\ M_2(\tau_0 + t) &= 4 - 3t \\ M_3(\tau_0 + t) &= 0 \\ M_4(\tau_0 + t) &= 2t \end{aligned}$$

   L'evento che fa evolvere la rete al prossimo IB-state è quello che si verifica quando si svuota il posto p_2.

2. IB-state 2 dopo un tempo pari a $t = \frac{4}{3}(\tau_1 = \tau_0 + t = \frac{4}{3})$ costituito dalla marcatura $M = (\frac{5}{3},\ 0,\ 0,\ \frac{8}{3})$ e da un vettore delle velocità istantanee $\vec{v} = (1,\ 0,\ 1)$. Le equazioni che regolano l'evoluzione a partire dal nuovo valore del tempo $\tau_1 = \frac{4}{3}$ sono:

$$\begin{aligned} M_1(\tau_1 + t) &= \tfrac{5}{3} - t \\ M_2(\tau_1 + t) &= 0 \\ M_3(\tau_1 + t) &= 0 \\ M_4(\tau_1 + t) &= \tfrac{8}{3} + 2t \end{aligned}$$

   L'evento che fa evolvere la rete al prossimo IB-state è quello che si verifica quando si svuota il posto p_1.

3. IB-state 3 dopo un tempo pari a $t = \frac{5}{3}(\tau_2 = \tau_1 + t = 3)$ costituito dalla marcatura $M_0 = (0,\ 0,\ 0,\ 1)$ e da un vettore delle velocità istantanee $\vec{v} = (0,\ 0,\ 0)$. Le equazioni che regolano l'evoluzione a partire dal dal nuovo valore del tempo $\tau_2 = 3$ sono:

$$\begin{aligned} M_1(\tau_2) &= 0 \\ M_2(\tau_2) &= 0 \\ M_3(\tau_2) &= 0 \\ M_4(\tau_2) &= 1 \end{aligned}$$

Non ci sono altri eventi che permettono l'evoluzione della rete: infatti il vettore delle velocità istantanee $\vec{v} = (0,\ 0,\ 0)$ fa si che il valore delle marcature nei posti rimanga costante.

3.2.2 Velocità minima e massima

Fino ad ora abbiamo parlato solamente di velocità massima, sott'intendendo che la velocità minima associata alle transizioni fosse nulla. Introduciamo il concetto di velocità minima associato ad una transizione, dando una nuova definizione di rete continua temporizzata.

Definizione 3.2.13. *Una CPN è caratterizzata dalla struttura algebrica $N_{cv} = (N_c, V)$ dove:*

- $N_c = (P, T, Pre, Post)$ *definita nella definizione ;*
- $V : T \longrightarrow \mathbb{R}_0^+ \times \mathbb{R}_\infty^+$ *è una funzione che, data una transizione j-esima mi restituisce la velocità minima e massima associata; V è una matrice $n_c \times 2$ dove $V(j,1)$ indica la velocità minima associata a t_j e $V(j,2)$ indica la velocità massima associata a t_j.*

Consideriamo l'esempio in Figura 3.4.

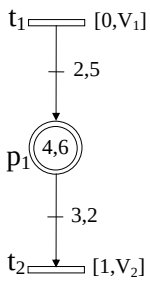


Figura 3.4: CPN con transizioni aventi una velocità minima e massima.

La transizione t_1 ha una velocità minima pari a 0 e una velocità massima pari a V_1, mentre t_2 ha una velocità minima pari a 1 e una velocità massima pari a V_2. D'ora in avanti, qualora specifichiamo solo la velocità massima associata ad una transizione, stiamo sott'intendendo che la velocità minima è nulla.

3.2.3 Evoluzione: problema di programmazione lineare

Fino ad ora abbiamo considerato l'evoluzione sott'intendendo sempre la massimizzazione della somma delle velocità istantanee delle transizioni, rispettando

i vincoli imposti dalla definizione 3.2.10; tuttavia possiamo voler scegliere, ad esempio, di annullare la velocità istantanea di alcune transizioni per massimizzare la quantità di fluido presente in un posto. Stiamo introducendo il concetto di *problema di programmazione lineare*: tutte le volte che vogliamo far evolvere una rete di Petri continua temporizzata applichiamo l'algoritmo 4.1.2, ogni qual volta si vuole passare da un IB-state al successivo IB-state, dove abbiamo indicato con C_1 e C_2 i vincoli che deve rispettare la funzione obiettivo J (spieghiamo l'algoritmo pari passo con un esempio mostrato in Figura 3.6).

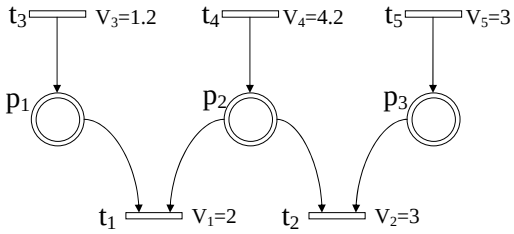


Figura 3.5: CPN con transizioni aventi una velocità minima e massima.

Algoritmo 3.2.1. *Evoluzione temporale di una rete di Petri continua:*

1. *Inizializzazione:* $M = M_0$.

2. Limite delle velocità. *In accordo con la definizione del modello, la velocità istantanea associata ad una transizione deve essere compresa fra la velocità minima e massima associata alla transizione stessa:*

$$C_1 = \{0 \leq v_1 \leq 2, 0 \leq v_2 \leq 3, 0 \leq v_3 \leq 1.2, 0 \leq v_4 \leq 4.2, 0 \leq v_5 \leq 3\}.$$

3. Bilanciamenti non negativi. *Poichè una marcatura non può essere negativa, il bilanciamento di un posto p_i che ha una marcatura $M(p_i) = 0$ all'inizio di un IB-state, non può essere negativo:*

$$C_1 = \{v_3 - v_1 \geq 0, v_4 - (v_1 + v_2) \geq 0, v_5 - v_2 \geq 0\}.$$

4. Calcolo delle velocità istantanee. *Consiste nel risolvere il seguente problema di programmazione lineare, massimizzando (o minimizzando) la seguente funzione obiettivo*

$$J = \sum_{t_j \in T} \beta_j \; v_j \text{ dove } \beta \in \mathbb{R}_0^+$$

I parametri β rappresentano i pesi associati alle transizioni, quindi, hanno il compito di assegnare un livello di priorità; avranno molta importanza per la risoluzione di eventuali conflitti attuali (si veda paragrafo 3.2.4). Con la risoluzione della funzione obiettivo otteniamo il valore del vettore $\vec{v}$. Nell'esempio supponiamo di voler massimizzare la quantità di marcatura contenuta in p_1, p_2, p_3: una possibile funzione obiettivo è

$$J = v_3 + v_4 + v_5.$$

5. *Calcolare la durata $t = \Delta t$ dell' IB-state (il minor tempo per il quale una componente di $M(\tau + t)$ diventa nulla); nell'esempio la marcatura nei posti tende a crescere sempre in quanto abbiamo annullato le velocità che li svuotano, quindi all'avanzare del tempo non passeremo mai ad un IB-state successivo (decido un tempo limite $t = time\_stop$ in cui voglio far terminare la simulazione).*

6. *Calcolare l'evoluzione della marcatura fino al tempo τ come indicato nella definizione 3.2.11*
$$M(\tau + t) = M(\tau) + C\ \vec{v}\ t$$

3.2.4 Conflitti e metodi di risoluzione

Se esiste un conflitto effettivo, il calcolo delle velocità istantanee richiede la risoluzione del conflitto. In questo paragrafo spiegheremo quando si verifica un conflitto effettivo e il metodo adottato per la sua risoluzione.

Definizione 3.2.14. *In una rete di Petri continua temporizzata c'è un **conflitto effettivo** al tempo $\tau = t$ se dato un sotto set $Q_j(\tau)$ di posti $p_i \in^\bullet t_j$ tali per cui $M(i) = 0$ risulta che*

1. $M(i) = 0$

2. $I_i(\tau) < \sum_{t_j \in p_i^\bullet} \min_{p_h \in Q_j(\tau)} (\frac{1}{Pre(p_h, t_j)} \sum_{t_k \in^\bullet p_h} Post(p_h, t_k) \cdot v_k(\tau),\ \ V_j)$

cioè se la velocità di riempimento dei posti p_k in conflitto strutturale non è sufficiente per lo scatto di tutte le transizioni in uscita alla velocità indicata nella Definizione 3.2.10. Nel caso di conflitto effettivo si ha che

$$v_j(\tau) \leq \min_{p_i \in Q_j(\tau)} (\frac{1}{Pre(p_i, t_j)} \sum_{t_k \in^\bullet p_i} Post(p_i, t_k) \cdot v_k(\tau),\ \ V_j)$$

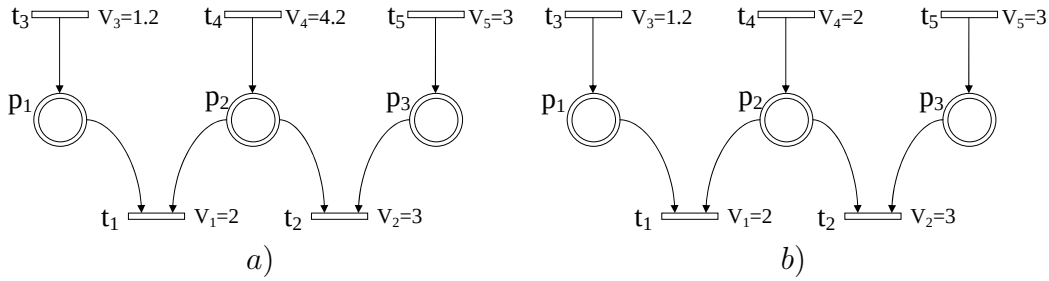


Figura 3.6: a) Nessun conflitto effettivo. b) Conflitto effettivo.

Esempio 3.2.3. *Consideriamo la rete in Figura 3.6 a): sia t_1 che t_2 sono debolmente abilitate e $M_2(\tau + t) = 0$ per $\tau = 0$. La velocità istantanea di t_1 è limitata dalle velocità di riempimento di p_1 e p_2 mentre quella di t_2 è limitata dalle velocità di riempimento di p_2 e p_3: segue che*

- $v_1 \leq \min(V_1, v_3, v_4) = v_3 = 1.2$
- $v_2 \leq \min(V_2, v_4, v_5) = v_5 = 3$

Verifichiamo ora se vi è o meno la presenza di conflitto effettivo cioè se $v_1 + v_2 \leq v_4$, che rappresentano le velocità istantanee che riguardano il posto p_2 in conflitto strutturale: in effetti $v_1 + v_2 = 1.2 + 3 = v_4$, quindi la velocità di riempimento v_4 del posto p_2 è sufficiente per far si che le sue velocità di svuotamento possano andare alla massima velocità consentita. Non ho la presenza di conflitto effettivo. Consideriamo, ora,la rete in Figura 3.6 b): t_1 e t_2 restano ancora debolmente abilitate e $M_2(\tau + t) = 0$ per $\tau = 0$. La velocità istantanea di t_1 è limitata dalle velocità di riempimento di p_1 e p_2 mentre quella di t_2 è limitata dalle velocità di riempimento di p_2 e p_3: segue che

- $v_1 \leq \min(V_1, v_3, v_4) = v_3 = 1.2$
- $v_2 \leq \min(V_2, v_4, v_5) = v_5 = 2$

Risulta evidente che la condizione $v_1 + v_2 \leq v_4$ non è più rispettata in quanto $v_1 + v_2 = 1.2 + 2 > v_4$, quindi la velocità di riempimento v_4 del posto p_2 non è più sufficiente a far si che le sue velocità di svuotamento possano andare alla massima velocità consentita: c'è la presenza di conflitto effettivo. Le velocità v_1 e v_2 possono andare ad una velocità tale per cui $v_1 + v_2 \leq v_4$.

Ma quale velocità istantanea attribuiamo a ciascuna transizione? Quando c'è un conflitto effettivo tra 2 o più transizioni, la tecnica principalmente utilizzata per la sua risoluzione si basa sull'assegnazione a ciascuna transizione di un peso che

inseriamo nella funzione obiettivo: quindi, l'algoritmo 4.1.2 è in grado, di per se, di risolvere gli eventuali problemi di conflitto effettivo grazie all'assegnazione dei pesi delle transizioni tramite i parametri β. Nel caso dell'esempio 3.2.3, se come funzione obiettivo impostiamo $J = 2v_1 + v_2 + v_3 + v_4 + v_5$.la transizione $t_1 > t_2$ cioè la transizione t_1 ha un peso maggiore sulla transizione t_2 ($\beta_1 > \beta_2$),quindi

- $v_1 = 1.2$
- $v_2 = v_4 - v_1 = 2 - 1.2 = 0.8$

La transizione t_1 può scattare alla massima velocità istantanea $v_1 = 1.2$ consentita in funzione dei vincoli sulla rete, mentre la transizione t_2, avendo un livello di priorità più basso, si deve adattare e deve abbassare la massima velocità istantanea da $v_2 = 2$ (non permessa) a $v_2 = 0.8$ (permessa). Invece, se come funzione obiettivo impostiamo $J = v_1 + 2v_2 + v_3 + v_4 + v_5$, la transizione $t_1 < t_2$ cioè la transizione t_1 ha un peso minore sulla transizione t_2 ($\beta_1 < \beta_2$),quindi

- $v_1 = 0$
- $v_2 = v_4 = 2$

La transizione t_1 non può più scattare mentre la transizione t_2, avendo un livello di priorità più alto, può scattare alla massima velocità istantanea consentita ($v_2 = 2$).

3.2.5 Buffer con capacità zero

Spesso è necessario imporre dei vincoli di sincronizzazione tra le transizioni continue. Ad esempio, se vogliamo ottenere che il flusso totale attraverso le transizioni t_1 e t_2 sia uguale al flusso attraverso le transizioni t_3, t_4 e t_5, stiamo richiedendo che

$$v_1 + v_2 \quad = \quad v_3 + v_4 + v_5$$

Questo lo possiamo ottenere introducendo i buffer a capacità nulla, rappresentati attraverso posti continui vuoti p_1 e p_2: consideriamo la Figura 3.7.

I posti vuoti p_1 e p_2 forzano 2 vincoli dati nella definizione 3.2.11 sulle velocità istantanee di scatto delle transizioni continue di ingresso e d'uscita da un posto,secondo cui:

$$\begin{cases} v_1 + v_2 & \geq & v_3 + v_4 + v_5 \\ v_3 + v_4 + v_5 & \geq & v_1 + v_2 \end{cases}$$

Si noti come il buffer introduca un ciclo vuoto, grazie al quale possiamo imporre degli ulteriori vincoli che ci permettono di forzare la scelta del vettore IFS nel problema di programmazione lineare.

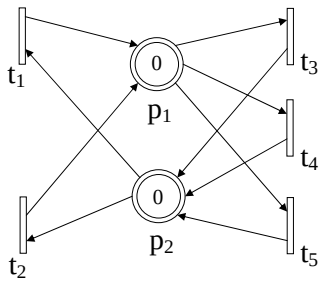


Figura 3.7: CPN con transizioni aventi una velocità minima e massima.

Capitolo 4

Reti di Petri ibride

Le reti di Petri ibride (HPN) vengono introdotte e descritte come il risultato dell'unione delle reti di Petri temporizzate e delle reti di Petri continue. In una HPN, pertanto, sono sempre presenti 2 parti, una continua e l'altra discreta, collegate da archi che uniscono un nodo (posto o transizione) discreto a uno continuo. Le 2 parti possono influenzarsi a vicenda, modificando i rispettivi stati. In questo capitolo verrà spiegato qual è la struttura di una rete di Petri ibrida e come essa evolve. In un modello ibrido utilizzo la parte discreta per differenziare la dinamica evolutiva

4.0.6 Struttura della rete

Definizione 4.0.15. *Una HPN è una struttura $N = (P, T, Pre, Post, Tempo, V)$ dove:*

- $P = P_c \bigcup P_d$ *è un insieme finito di* m posti *non vuoto; P è diviso in un sotto insieme di posti continui P_c e in un sotto insieme di posti discreti P_d di cardinalità rispettivamente mc e md.*

- $T = T_c \bigcup T_d$ *è un insieme finito di* n transizioni *non vuoto; T è diviso in un sotto insieme di posti continui T_c e in un sotto insieme di posti discreti T_d di cardinalità rispettivamente nc e nd.*

- Pre *è la* funzione di pre-incidenza *che specifica gli archi diretti dai posti alle transizioni (detti archi Pre) e viene rappresentata mediante una matrice $m \times n$; più precisamente, $Pre(p_i, t_j)$ indica quanti archi vanno dal posto p_i alla transizione t_j: in particolare*

$$Pre : \begin{cases} P_c \times T \to \mathbb{R}_0^+ \\ P_d \times T \to \mathbb{N} \end{cases}$$

- $Post$ *è* funzione di post-incidenza *che specifica gli archi diretti dalle transizioni ai posti (detti archi* $Post$*) e viene rappresentata mediante una matrice* $m \times n$*; più precisamente,* $Post(p_i, t_j)$ *indica quanti archi vanno dalla transizione* t_j *al posto* p_i*: in particolare*

$$Post : \begin{cases} P_c \times T \to \mathbb{R}_0^+ \\ P_d \times T \to \mathbb{N} \end{cases}$$

- *Tempo:* $T_d \longrightarrow \mathbb{R}_0^+$ *è una funzione associata all'insieme di transizioni* T_d *secondo la quale* $Tempo(t_j) = d_j =$ *tempo associato a* t_j*;*

- $V : T_c \longrightarrow \mathbb{R}_0^+ \times \mathbb{R}_\infty^+$ *è una funzione che, data una transizione* $t_j \in T_c$ *restituisce la velocità minima e massima associata;* V *è una matrice* $n_c \times 2$ *dove* $V(j,1)$ *indica la velocità minima associata a* t_j *e* $V(j,2)$ *indica la velocità massima associata a* t_j*.*

Imponiamo che per tutte le $t_j \in T_c$ e per tutti i $p_i \in P_d$, $Pre(p_i, t_j) = Post(p_i, t_j)$ cioè lo scatto delle transizioni continue non cambia la marcatura dei posti discreti. Assumiamo che i posti continui e discreti in ingresso alla transizione $t_j \in T$ siano rispettivamente ${}^{(c)}t_j =^{\cdot} t \bigcap P_c$ e ${}^{(d)}t_j =^{\cdot} t \bigcap P_d$; stesso formalismo può essere adottato con le transizioni continue e discrete in ingresso a un posto $p_i \in P$.

Definizione 4.0.16. *Data una rete* N*, con* m *posti ed* n *transizioni, definisco la* matrice di incidenza

$$C : \begin{cases} P_c \times T \to \mathbb{R}_0^+ \\ P_d \times T \to \mathbb{N} \end{cases}$$

cioè il generico elemento di C *vale* $C(p_i, t_j) = Post(p_i, t_j) - Pre(p_i, t_j)$*.*

La restrizione di Pre, $Post$ e C a P_X e T_Y ($X, Y \in \{c, d\}$) è denotata rispettivamente Pre_{XY}, $Post_{XY}$ e C_{XY}.

Esempio 4.0.4. *Consideriamo la rete in Figura 4.2: il posto* p_1 *è continuo; i posto* $, p_3, p_4, p_5$ *sono discreti. Le transizioni* t_1 *e* t_2 *sono transizioni continue che hanno entrambe velocità minima* 0 *e che hanno velocità massima rispettivamente* V_1 *e* V_2*; assumiamo che* $V_1\ a < V_2\ b$ *(dove* a *e* b *sono i pesi degli archi dati da* Pre *e* $Post$*). Le transizioni discrete* t_3, t_4, t_5, t_6 *sono temporizzate distribuite esponenzialmente la cui frequenza di scatto è rispettivamente* $\lambda_3, \lambda_4, \lambda_5, \lambda_6$*. Le* 2 *transizioni continue rappresentano* 2 *macchine inaffidabili che si possono guastare con tasso* λ_3 *e* λ_5 *e possono essere riparate con tasso* λ_4 *e* λ_6*: le parti prodotte dalla prima macchina (*t_1*) sono poste in un buffer (posto* p_1*) e poi vengono lavorate dalla seconda macchina (*t_2*). L'ampiezza dell'arco* a *rappresenta il rapporto*

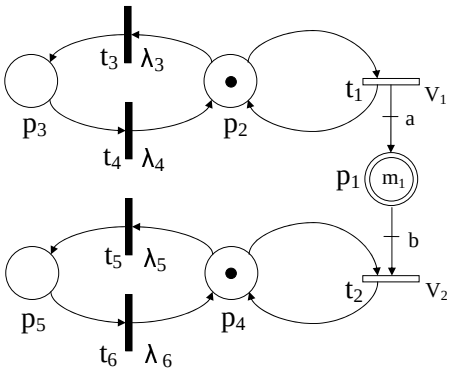


Figura 4.1: Sistema di produzione attraverso 2 macchine: modello HPN equivalente.

fra il flusso lavorato dalla prima macchina e il flusso messo nel buffer; l'ampiezza dell'arco b rappresenta il rapporto fra il flusso lavorato dalla seconda macchina e il flusso preso dal buffer. La matrice di incidenza di questa rete è

$$C = \begin{bmatrix} C_{cc} & C_{cd} \\ C_{dc} & C_{dd} \end{bmatrix} = \left[\begin{array}{cc|cccc} a & -b & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{array}\right]$$

4.0.7 Marcatura e abilitazione

Mediante la marcatura è possibile definire lo stato di una HPN.

Definizione 4.0.17. *Una* marcatura

$$M = \begin{cases} P_c \longrightarrow \mathbb{R}_0^+ \\ P_d \longrightarrow \mathbb{N} \end{cases}$$

è una funzione che assegna ad ogni posto continuo un volume di fluido ed a ogni posto discreto un numero intero non negativo di gettoni.

Indicheremo con M^c la marcatura relativa ai posti $p_i \in P_c$ e con M^d la marcatura relativa ai posti $p_i \in P_d$. Ne segue che $M = M^c \bigcup M^d$.

Definizione 4.0.18. *Una rete N con una marcatura iniziale M_0 è detta* rete marcata o sistema di rete *e viene indicata come $R = \langle N, M_0 \rangle$.*

Si noti che il vettore della generica marcatura M è un vettore colonna $m \times 1$,quindi ha tante righe quanti sono i posti P della rete.

L'abilitazione di una *transizione discreta* dipende dalla marcatura di tutti i posti in ingresso, sia discreti che continui.

Definizione 4.0.19. *Data una rete di Petri ibrida $\langle N, M_0 \rangle$, una transizione discreta $t_j \in T_d$ è abilitata a partire dalla marcatura M se*

$$M(p_i) \geqslant Pre(p_i, t_j) \quad p_i \in^{\bullet} t_j$$

L'abilitazione di una *transizione continua* dipende dalla marcatura dei soli posti discreti in ingresso. La marcatura dei posti continui in ingresso a una transizione continua è utilizzata per distinguere se essa è fortemente o debolmente abilitata.

Definizione 4.0.20. *Data una rete di Petri ibrida $\langle N, M_0 \rangle$, una transizione continua $t_j \in T_c$ è abilitata a partire dalla marcatura M se*

$$M^d(p_i) \geqslant Pre(p_i, t_j) \quad p_i \in^{(d)} t_j$$

Una transizione $t_j \in T_c$ *si dice*

- ***fortemente abilitata*** *al tempo τ se*

$$M^c(p_i) > 0 \quad \forall p_i \in^{(c)} t_j$$

- ***debolmente abilitata*** *al tempo τ se*

$$M^c(p_i) = 0 \quad \forall p_i \in^{(c)} t_j$$

Esempio 4.0.5. *Nella rete in Figura 4.2, la parte discreta rappresenta il modello di guasto delle macchine. Quando il posto p_2 è marcato la transizione t_1 è abilitata e la prima macchina è operativa; quando il posto p_3 è marcato la transizione t_1 non è abilitata e la prima macchina è ferma. Una interpretazione simile è applicata anche alla seconda macchina. La marcatura rappresentata nella rete mostra che, inizialmente, entrambe le macchine sono in funzione e il buffer contiene un fluido pari a m_1. La transizione t_1 è fortemente abilitata e così come la transizione t_2, fin tanto che $m_1 > 0$.*

Definizione 4.0.21. *Data una marcatura M si definisce grado di abilitazione della transizione $t_j \in T_d$*

$$enab(M, t_j) = \max\{k \in \mathbb{N} : \quad M \geq k\ Pre(\cdot, t_j)\}$$

allora t_j è detta essere $enab(M, t_j)$ volte abilitata.

Possiamo definire un nuovo vettore riga e_{ab} di dimensione $1 \times m$, dove:

- se $t_j \in T_c$ allora $e_{ab}(j) = \{0, 1\}$ a seconda che la transizione sia abilitata o non abilitata;
- se $t_j \in T_d$ allora $e_{ab}(j) = enab(M, t_j)$ è uguale al grado di abilitazione per la transizione j-esima.

4.1 Dinamiche della rete

Ci apprestiamo a descrivere le dinamiche di una HPN: prima considereremo il comportamento associato allo scatto delle transizioni discrete e poi quello associato alle transizioni continue.

4.1.1 Dinamiche delle transizioni discrete

Applichiamo l'algoritmo 2.2.1 sostituendo solamente il passo 9) col passo seguente:

- Il tempo avanza fino all'istante $\tau_{l+1} = \tau_l + o^*$ e il sistema si evolve giungendo alla nuova marcatura

$$\begin{cases} M^c_{l+1}(\tau_{l+1}) = M^c_l(\tau_l) + C_{cd} \cdot \vec{\sigma} \\ M^d_{l+1}(\tau_{l+1}) = M^d_l(\tau_l) + C_{dd} \cdot \vec{\sigma} \end{cases}$$

4.1.2 Dinamiche delle transizioni continue

Consideriamo l'algoritmo e adattiamolo alle reti di Petri ibribe

Algoritmo 4.1.1. *Evoluzione temporale di una rete di Petri ibrida dovuta alle sole transizioni continue:*

1. *Inizializzazione:* $M = M_0$.
2. Limite delle velocità. *In accordo con la definizione del modello, la velocità istantanea associata ad una transizione deve essere compresa fra la velocità minima e massima associata alla transizione stessa; si impongono i vincoli* C_1 *solo sulle transizioni* $t_j \in T_c$ *per cui* $e_{ab}(j) = 1$.
3. Bilanciamenti non negativi. *Poichè una marcatura non può essere negativa, il bilanciamento di un posto* $p_i \in P_c$ *che ha una marcatura* $M(p_i) = 0$ *all'inizio di un IB-state, non può essere negativo; si impongono i vincoli* C_2 *solo sulle transizioni* $t_j \in T_c$ *per cui* $e_{ab}(j) = 1$.

4. Calcolo delle velocità istantanee. *Consiste nel risolvere il seguente problema di programmazione lineare, massimizzando (o minimizzando) la seguente funzione obiettivo*

$$J = \sum_{t_j \in T} \beta_j \, v_j \quad \textit{dove} \quad \beta \in \mathbb{R}^+$$

I parametri β rappresentano i pesi associati alle transizioni, quindi, hanno il compito di assegnare un livello di priorità. Con la risoluzione della funzione obiettivo otteniamo il valore del vettore $\vec{v}$.

5. *Calcolare la durata $t = \Delta t$ dell' IB-state (il minor tempo per il quale una componente di $M(\tau + t)$ diventa nulla);*

6. *Calcolare l'evoluzione della marcatura fino al tempo $\tau + t$ come indicato nella definizione 3.2.11*

$$\begin{cases} M^c(\tau + t) = M^c(\tau) + C_{cc} \cdot \vec{v} \cdot t \\ M^d(\tau + t) = M^d(\tau) \end{cases}$$

Considerando l'esempio in Figura 4.2: se $M(p_1) > 0$ allora avremo solamente il vincolo $C_1 = \{0 \leqslant v_1 \leqslant V_1, \quad 0 \leqslant v_2 \leqslant V_2\}$; se scegliamo come funzione obiettivo $J = v_1 + v_2$ allora, fin tanto che t_1 e t_2 rimangono abilitate, otteniamo che $v_1 = V_1$ e $v_2 = V_2$. Se $M(p_1) = 0$ allora dobbiamo inserire, oltre al vincolo C_1, anche il vincolo $C_1 = \{a \, v_1 \leqslant b \, v_2\}$; mantenendo la stessa funzione obiettivo $J = v_1 + v_2$, fin tanto che t_1 e t_2 rimangono abilitate, otteniamo che $v_1 = v_2 = \min(a \, V_1, b \, V_2)$.

4.2 Comportamento macro: un modello lineare

In questa sezione mostreremo come sia possibile combinare la dinamica continua con quella discreta. Prima daremo una definizione dei macro eventi e macro periodi di una HPN e poi analizzeremo le possibili dinamiche di evoluzione.

4.2.1 Macro eventi e macro periodi

L'evoluzione del sistema è guidata da 4 tipi di macro eventi:

- π_i: un posto continuo p_i si svuota. Questo evento cambia il grado di abilitazione di un insieme di transizioni continue da fortemente abilitate a debolmente abilitate, modificando i vincoli inseriti in C_2.

- γ_j: una transizione discreta t_j scatta. Questo evento cambia la marcatura discreta e può abilitare/disabilitare un insieme di transizioni continue oppure può abilitare/disabilitare un insieme di transizioni discrete modificando, quindi, la matrice degli orologi Q.
- ε_i: la marcatura di un posto continuo p_i aumenta, raggiungendo un livello di fluido che abilita un insieme di transizioni discrete e abilitando i corrispondenti orologi nella matrice Q.
- $\overline{\varepsilon}_i$: la marcatura di un posto continuo p_i decresce, raggiungendo un livello di fluido che disabilita un insieme di transizioni discrete e disabilitando i corrispondenti orologi nella matrice Q.

Definiamo τ_k, per $k = 0, 1, 2, \cdots$, come il tempo in cui si verifica il k- esimo macro evento. L'intervallo $[\tau_k, \tau_{k+1}]$ è chiamato macro periodo e lo indichiamo come $\Delta(k+1) = (\tau_{k+1} - \tau_k)$: si noti come un macro periodo possa avere una durata nulla se scatta una transizione discreta immediata.

4.2.2 Durata dei macro periodi

Una rete di Petri ibrida evolve da un macro periodo in un altro macro periodo quando si verificano i macro eventi. Abbiamo visto che ci sono 4 tipi di macro evento, ciascuno dei quali modifica in modo diverso l'evoluzione della rete; calcoliamo la durata di un macro evento:

- *Macro evento* π_i. La lunghezza del macro periodo è il tempo necessario affinchè il posto continuo p_i si svuoti, cioè è il rapporto fra la marcatura attuale e la sua variazione rispetto al tempo (cambiata di segno):
$$\Delta(k+1) = \frac{-M(p_i, k)}{\dot{M}(p_i, k)} = \frac{-M(p_i, k)}{C_{cc} v(k)}$$
- *Macro evento* γ_i. Se $t_j \in T_d$, la lunghezza del macro periodo è il tempo residuo di scatto della transizione t_j:
$$\Delta(k+1) = o^* = \min_{t_j \in T_d} \{o_i\}$$
Si veda l'algoritmo 2.2.1 per maggiori dettagli su come calcolare gli orologi associati alle transizioni discrete.
- *Macro evento* ε_i, $\overline{\varepsilon}_i$. La lunghezza del macro periodo è il tempo necessario affinchè la marcatura $M(p_i)$ raggiunga il valore $C(p_i, t_j)$, abilitando (o disabilitando) la transizione discreta t_j:
$$\Delta(k+1) = \frac{C(p_i, t_j) - M(p_i, k)}{\dot{M}(p_i, k)} = \frac{C(p_i, t_j) - M(p_i, k)}{C_{cc} v(k)}$$

4.3 Algoritmo di evoluzione

In questo paragrafo verrà fornito un algoritmo di evoluzione per determinare il macro stato tra un macro periodo e il successivo macro periodo: un macro stato k-esimo al tempo τ, $\mathbf{x}_k(\tau)$, è costituito dal valore della marcatura $M_k(\tau) = \left[\begin{array}{cc} M^c(\tau), & M^d(\tau) \end{array} \right]^T$

Algoritmo 4.3.1. *Algoritmo di evoluzione.*

1. *Sia $k = 0$, $\tau = 0$.*

2. *Scegliere una funzione obiettivo J per il calcolo delle velocità istantanee delle transizioni continue e impostare il tempo massimo $time\_stop$ di evoluzione della rete.*

3. *Calcolare il valore della marcatura corrente $M_k(\tau)$ e il vettore di abilitazione $e_{ab,k}$.*

4. *Calcolare la matrice degli orologi Q associati alle transizioni discrete (seguire algoritmo 2.2.1 passo 7).*

5. *Calcolare i vincoli C_1 e C_2 relativi alle velocità delle transizioni continue.*

6. *Calcolare il vettore delle velocità istantanee di scatto $\vec{v}$ in base a J, C_1 e C_2*

7. *Determinare il prossimo macro evento $\overline{\eta}$ in accordo con i seguenti passi:*

   (a) *Sia $\Psi_k = \emptyset$ (questo insieme contiene tutte le coppie (η, Δ_η) dove η è un evento che può potenzialmente verificarsi e Δ_η e il suo tempo residuo di avvenimento.)*

   (b) *Se $o^* = 0$ allora può scattare una transizione $t_j \in T_d$ immediata: aggiungo a Ψ_k la coppia $(\gamma_i, 0)$.*

   (c) *Se $\Psi_k \neq \emptyset$ vai al passo 8, altrimenti prosegui.*

   (d) *Per ogni transizione $t_j \in T_d$ abilitata da $M_k(\tau)$, calcolare la transizione t_j che possiede uno o più orologi $o_i = o^*$ (si veda algoritmo 2.2.1 passo 8) e aggiungere a Ψ_k la coppia (γ_i, o^*).*

   (e) *Per ogni posto continuo $p_i \in P_c$ non vuoto, se $\dot{M}(p_i, k) = C_{cc}v(k) < 0$ allora aggiungere a Ψ_k la coppia $(\pi_i, \frac{-M(p_i,k)}{C_{cc}v(k)})$.*

(f) Per ogni transizione $t_j \in T_d$ non abilitata da $M_k(\tau)$, sia $P_i = \{p_l \in^{(c)} t_j | M_l(k) < C(p_l, t_j)\}$ l'insieme dei posti continui che hanno abbastanza fluido per abilitare t_j. Questa transizione può abilitarsi alla fine del corrente macro periodo k se le seguenti condizioni sono entrambe verificate:

- *$M^d(k) \geq C_{dd}(\cdot, t_j)$, cioè t_j è abilitata nella sotto rete discreta;*
- *$\forall p_l \in P_i, \dot{M}_l(k) = C_{cc} v(k) > 0$, cioè la marcatura di tutti i posti P_i è aumentata.*

Il tempo dopo il quale t_j si abilita è

$$\Delta = \max p_l \in P_i \frac{C(p_l, t_j) - M_l(k)}{\dot{M}_l(k)}$$

In questo caso indichiamo con p_i il posto per il quale questo valore è massimo e dovremo aggiungere a Ψ_k la coppia (ε_i, Δ).

(g) Per ogni transizione $t_j \in T_d$ abilitata da $M_k(\tau)$, sia $\bar{P}_i = \{p_l \in^{(c)} t_j | \dot{M}_l(k) < 0\}$ l'insieme dei posti continui il cui valore della marcatura sta diminuendo. Se $\bar{P}_i \neq \emptyset$, la transizione t_j può disabilitarsi dopo un tempo pari a

$$\Delta = \min p_l \in \bar{P}_i \frac{C(p_l, t_j) - M_l(k)}{\dot{M}_l(k)}$$

In questo caso indichiamo con p_i il posto per il quale questo valore è minimo e dovremo aggiungere a Ψ_k la coppia $(\bar{\varepsilon}_i, \Delta)$.

(h) Scegliamo da Ψ_k la coppia $(\bar{\eta}, \Delta_{\bar{\eta}})$ per cui $\Delta_{\bar{\eta}}$ è il minimo fra tutte le coppie. L'evento $\bar{\eta}$ è il prossimo evento che si verifica. Se $\Psi_k = \emptyset$ allora $\Delta(k+1) = time\_stop$ dove $time\_stop$ è il tempo massimo di evoluzione che è stato impostato al passo 2. Nel caso in cui ci sia un conflitto generale fra le transizioni discrete, si scelgono quelle transizioni che devono scattare in base a un algoritmo di risoluzione dei conflitti (stocastico 2.2.2 o deterministico 2.2.3) e le si memorizzano nel vettore di scatto $\vec{\sigma}$.

8. *Calcolare il valore della marcatura del macro periodo $k+1$ in funzione degli eventi $\bar{\eta}$ che si sono verificati:*

$$\begin{cases} M^c(k+1) = M^c(k) + C_{cc} \cdot \vec{v} \cdot t + C_{cd} \cdot \vec{\sigma} \\ M^d(k+1) = M^d(k) + C_{dd} \cdot \vec{\sigma} \end{cases}$$

9. *Aggiornare il valore degli orologi di $Q(k+1) = Q(k) - \Delta_{\bar{\eta}}$.*

10. $k = k + 1$.

11. Ripeti dal passo 3).

12. Fine dell'evoluzione.

Esempio 4.3.1. *Consideriamo la rete in Figura 4.2 L'evoluzione viene fatta ini-*

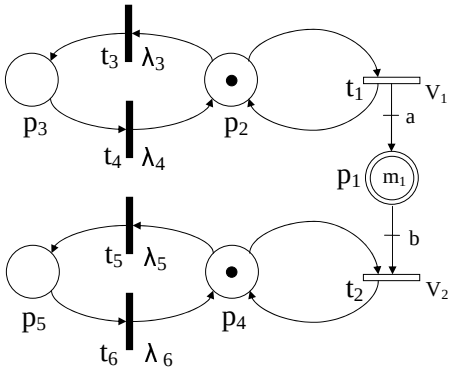


Figura 4.2: Sistema di produzione: modello HPN equivalente.

ziare al tempo $\tau = 0$*; assumiamo di voler massimizzare l'utilizzo di entrambe le macchine, cioè* $J = v_1 + v_2$.

$k = 0$**.** *La marcatura iniziale vale* $M_0 = [m_1, 1, 0, 1, 0]^T$*. Le transizioni stocastiche abilitate sono* t_3 *e* t_5*: poichè il grado di abilitazione di entrambe è pari a* 1*, estraiamo da ogni distribuzione un solo valore cioè* $\hat{d}_3$ *per la transizione* t_3 *e* $\hat{d}_5$ *per la transizione* t_5*; supponiamo che* $\hat{d}_5 < \hat{d}_3$*. Le transizioni* t_1 *e* t_2 *sono fortemente abilitate e possono scattare alla velocità massima, in accordo con la funzione obiettivo* J*:* $\vec{v} = [V_1, V_2]$*. Ricaviamo che* $\Psi_0 = \{(\pi_1, \Delta_{\pi 1}), (\gamma_3, \Delta_{\gamma 3}), (\gamma_5, \Delta_{\gamma 5})\}$*, dove* $\Delta_{\pi 1} = \frac{m_1}{(V_2 b - V_1 a)}$*,* $\Delta_{\gamma 3} = \hat{d}_3$*,* $\Delta_{\gamma 5} = \hat{d}_5$*. Assumiamo che* $\Delta_{\pi 1} < \hat{d}_5$*: il prossimo evento che si verifica è* π_1*, quindi il primo macro evento è dovuto allo svuotamento del posto continuo* p_1*. La lunghezza del primo macro periodo è* $\Delta(1) = \Delta_{\pi 1}$ *e il tempo in cui si verifica il prossimo macro evento è* $\tau = \tau_1 = \Delta_{\pi 1}$*. Il vettore di scatto* $\vec{\sigma} = [0, 0, 0, 0]$*; il nuovo valore della marcatura in funzione dell'evento* p_1 *sarà*

$$\begin{cases} M^c(1) = M^c(0) + C_{cc} \cdot \vec{v} \cdot \tau_1 = [0] \\ M^d(1) = M^d(0) + C_{dd} \cdot \vec{\sigma} = M^d(0) = [1, 0, 1, 0]^T \end{cases}$$

Aggiorno la matrice degli orologi $Q(1) = Q(0) - \tau_1$.

$k = 1$. *La marcatura attuale vale* $M_1 = [0, 1, 0, 1, 0]^T$. *Le transizioni stocastiche abilitate sono sempre* t_3 *e* t_5*: poichè il grado di abilitazione di entrambe è pari a* 1*, lasciamo inalterata la matrice* Q*. Le transizioni* t_1 *è fortemente abilitata mentre* t_2 *è debolmente abilitata. In accordo con la funzione obiettivo* J*:* $\vec{v} = [V_1, V_1\frac{a}{b}]$*. Ricaviamo che* $\Psi_1 = \{(\gamma_3, \Delta_{\gamma 3}), (\gamma_5, \Delta_{\gamma 5})\}$*, dove* $\Delta_{\gamma 3} = \hat{d}_3 - \tau_1$*,* $\Delta_{\gamma 5} = \hat{d}_5 - \tau_1$*. Il prossimo evento che si verifica è* γ_5*, quindi il prossimo macro evento è dovuto allo scatto di* t_5*. La lunghezza del secondo macro periodo è* $\Delta(2) = \Delta_{\gamma 5}$ *e il tempo in cui si verifica il prossimo macro evento è* $\tau_2 = \tau_1 + \Delta_{\gamma 5}$*. Il vettore di scatto* $\vec{\sigma} = [0, 0, 0, 1]$*; il nuovo valore della marcatura in funzione dell'evento* $\gamma 5$ *sarà*

$$\begin{cases} M^c(2) = M^c(1) + C_{cc} \cdot \vec{v} \cdot \Delta(2) = [0] \\ M^d(2) = M^d(1) + C_{dd} \cdot \vec{\sigma} = M^d(0) = [1, 0, 0, 1]^T \end{cases}$$

Aggiorno la matrice degli orologi $Q(2) = Q(1) - \Delta_{\gamma 5}$.

$k = 2$. *La marcatura attuale vale* $M_1 = [0, 1, 0, 0, 1]^T$. *Le transizioni stocastiche abilitate sono sempre* t_3 *e* t_6*: poichè il grado di abilitazione di* t_6 *è passato da* 0 *a* 1*, introduco nella matrice* Q *il timer associato alla transizione stessa (estraggo dalla distribuzione di* t_6 *un solo valore cioè* $\hat{d}_6$ *) ed elimino il timer di* t_5 *che si è disabilitata. Supponiamo che* $\hat{d}_3 - \tau_2 < \hat{d}_6$*. In accordo con la funzione obiettivo* J*:* $\vec{v} = [V_1, 0]$*. Ricaviamo che* $\Psi_2 = \{(\gamma_3, \Delta_{\gamma 3}), (\gamma_6, \Delta_{\gamma 6})\}$*, dove* $\Delta_{\gamma 3} = \hat{d}_3 - \tau_2$*,* $\Delta_{\gamma 6} = \hat{d}_6$*. Il prossimo evento che si verifica è* γ_3*, quindi il prossimo macro evento è dovuto allo scatto di* t_3*. La lunghezza del terzo macro periodo è* $\Delta(3) = \Delta_{\gamma 3}$ *e il tempo in cui si verifica il prossimo macro evento è* $\tau_3 = \tau_2 + \Delta_{\gamma 3}$*. Il vettore di scatto* $\vec{\sigma} = [1, 0, 0, 0]$*; il nuovo valore della marcatura in funzione dell'evento* $\gamma 3$ *sarà*

$$\begin{cases} M^c(3) = M^c(2) + C_{cc} \cdot \vec{v} \cdot \Delta(3) = [V_1 \Delta(3)] \\ M^d(3) = M^d(2) + C_{dd} \cdot \vec{\sigma} = M^d(0) = [0, 1, 0, 1]^T \end{cases}$$

L'algoritmo può proseguire ancora, ma ci blocchiamo qui poichè il nostro scopo era quello di far vedere come si applicava.

Capitolo 5

Il simulatore HYPENS

5.1 Introduzione

Il simulatore, che presenteremo in questo capitolo, permette di elaborare qualsiasi rete di Petri temporizzata fornitagli in ingresso, che può essere discreta (cioè con posti discreti e transizioni sia deterministiche che stocastiche), continua (cioè con posti continui e transizioni continue aventi velocità variabili costanti a tratti) oppure ibrida (risultato dell'unione delle reti di Petri temporizzate e delle reti di Petri continue). L'acronimo **HYPENS** sta per **Hy**brid **Pe**tri **N**ets **S**imulator, per sottolineare il fatto che si è creato un qualcosa in grado di simulare tutti i tipi di reti di Petri, siano essi discreti, continui o ibridi.
Il software è costituito da 4 file:

1. *enter_HPN.m* (interfaccia)
2. *make_HPN.m* (interfaccia)
3. *simulator_HPN.m* (simulatore)
4. *analysis_HPN.m* (analisi della simulazione)

E' stato realizzato sia in lingua italiana che in lingua inglese, diversificando, quindi, la visualizzazione di tutti i commenti a video a seconda della scelta iniziale.

In particolare, i primi 2 file si occupano di creare la rete e strutturarla in modo tale che, il terzo file (che rappresenta il simulatore vero e proprio) riesca ad interpretare in modo corretto i dati forniti dall'utente; il file $simulator\_HPN.m$ fornisce in uscita l'evoluzione temporale del sistema che viene memorizzata in una matrice di output denominata $Evol$ grazie alla quale possiamo fare sia delle analisi che delle rappresentazioni grafiche (quarto file). L'utilizzo dei primi 2 file

è a discrezione dell'utente il quale può decidere o di creare la rete, passo per passo, rispondendo a delle domande esplicite visualizzate a video (primo file) oppure può inserire i dati direttamente nell'input della $function$ (secondo file).

5.2 Il linguaggio di programmazione Matlab

Il Matlab (Matrix laboratory) è un linguaggio di programmazione matematico ad altissimo livello orientato alla manipolazione delle matrici e dei vettori. Le istruzioni fondamentali che possono essere usate sono in diretta corrispondenza delle funzioni che vengono usate normalmente in matematica per il calcolo matriciale e vettoriale (operazioni algebriche, esponenziale, inversa) e tutte le composizioni di queste funzioni possono essere rapidamente assemblate in una espressione di facile comprensione.

Ogni variabile, matrice o vettore è rappresentata in Matlab da una lettera o da una combinazione di lettere e numeri che gli viene associata. Ogni funzione complessa (composizione di funzioni semplici, così come il fattoriale di un numero è composizione di più moltiplicazioni) è facilmente realizzabile e memorizzabile per l'uso futuro o per l'utilizzazione in funzioni più complesse. Le funzioni complesse di uso più comune sono già disponibili nel Matlab come facenti parte di Toolbox, ovvero librerie dedicate ad applicazioni matematiche specialistiche (Signal Processing, Controllo di Processi, Statistica ecc.).

Due particolarità del Matlab rispetto ai linguaggi di programmazione più diffusi sono la possibilità di utilizzarlo con la linea di comando in una finestra di Windows detta Command Window, che consente di eseguire i comandi su una serie di variabili inserendoli direttamente da una tastiera come se si usasse una calcolatrice scientifica; l'altra particolarità è la presenza di un ambiente grafico integrato nel quale possono essere disegnati semplicemente gli andamenti delle funzioni che si vogliono analizzare.

Si è deciso di utilizzare Matlab perchè è un linguaggio di alto livello che si è imposto in ambito ingegneristico mondiale come efficace strumento di calcolo ed elaborazione. Il simulatore per le reti di Petri **HYPENS** è stato interamente creato ed utilizzato nella versione 7.1 di Matlab; quando si è andato a testare su versioni precedenti di Matlab ci si è resi conto che alcune funzioni rendevano un messaggio di errore: ad esempio la creazione di una matrice di not-a-number tramite il comando $NaN(x, y)$, dove x è il numero di righe e y è il numero di colonne, non era permesso. Si sono raggiarati tali limiti con alcuni stratagemmi che, nonostante non siano eleganti dal punto di vista dello stile del codice, sono altamente funzionali e hanno permesso di poter utilizzare il simulatore indipendentemente dalla versione di Matlab sul quale si manda in esecuzione.

5.3 HYPENS: un simulatore diverso dagli altri esistenti

Nell'arco degli ultimi anni sono stati creati diversi simulatori in grado di analizzare l'evoluzione e di calcolare diverse proprietà delle reti di Petri temporizzate e non temporizzate, delle reti di Petri continue e delle reti di Petri ibride. Andiamo ad elencare i principali e più completi simulatori in grado di analizzare almeno le reti di Petri temporizzate:

- **ARP**: E' un software per le reti di Petri in grado di analizzare reti di Petri con transizioni temporizzate aventi un intervallo temporale di scatto $[d_{jmin}$, $d_{jmax}]$; poi può estrapolare dati statistici quali valor medio della marcatura nei posti e frequenza di scatto delle transizioni. La simulazione avviene manualmente ste-by-step: il software è scritto in *Turbo Pascal 6*.

- **HISIm**: E' un software che permette di creare e simulare le reti di Petri ibride; è possibile visualizzare graficamente un'animazione dell'evoluzione della rete. Il simulatore è stato scritto in *Java*.

- **HPSim**: E' un simulatore che permette, attraverso l'ausilio di un editor grafico, di editare e simulare le reti di Petri temporizzate sia deterministiche che stocastiche. Il software è stato scritto in *Java*.

- **Petri Net Toolbox**:E' un software per la simulazione basato sulle reti di Petri; sono accettate reti con transizioni non temporizzate e temporizzate siano esse deterministiche che stocastiche e reti con posti temporizzati; i posti possono avere una capacità finita o infinita e i conflitti tra le transizioni possono essere risolti grazie all'assegnazione di priorità e probabilità di scatto. Tutto funziona tramite interfaccia grafica: il software è stato scritto in *Matlab*.

- **SIRPHYCO**: E' un software per la simulazione basato sulle reti di Petri; sono accettate reti con transizioni temporizzate sia deterministiche che stocastiche con distribuzione esponenziale; i posti possono avere una capacità finita o infinita e i conflitti tra le transizioni possono essere risolti grazie all'assegnazione di priorità; inoltre il simulatore è in grado di simulare reti di Petri continue ed ibride. Tutto funziona tramite interfaccia grafica: il software è stato scritto in *C++*.

Se si analizzano le caratteristiche dei simulatori elencati, si può notare come solo due sono in grado di simulare le reti di Petri ibride (HISIm e SIRPHYCO) e come solo uno sia stato sviluppato con il linguaggio Matlab (Petri Net Toolbox). Il

simulatore più completo fre quelli elencati è senza dubbio SIRPHYCO, anche se presenta diversi limiti:

1. Permette di simulare reti in cui le transizioni stocatische possono avere solo una distribuzione esponenziale: questa è un vincolo molto restrittivo se si vogliono simulare modelli che si attengono alla realtà! Se si vuole simulare un sistema in cui le transizioni hanno un andamento diverso da quello deterministico o stocastico esponenziale, non si può fare!

2. Non è possibile attribuire alle transizioni continue una velocità minima diversa da 0;

3. Se si vuole esplicitare un numero di serventi k diverso da 0 o ∞, si è costretti a indicarli nel modello con dei posti connessi tramite cappio alla transizione che si vuole limitare;

4. Se un posto discreto è in conflitto strutturale con una o più transizioni continue e una o più transizioni discrete, le transizioni discrete hanno sempre priorità sulle transizioni continue.

5. Il fatto che ci sia solo l'interfaccia grafica limita notevolmente l'utilizzo del software per la simulazione di reti con un numero elevato di posti e transizioni: basti vedere la Figura 5.1.

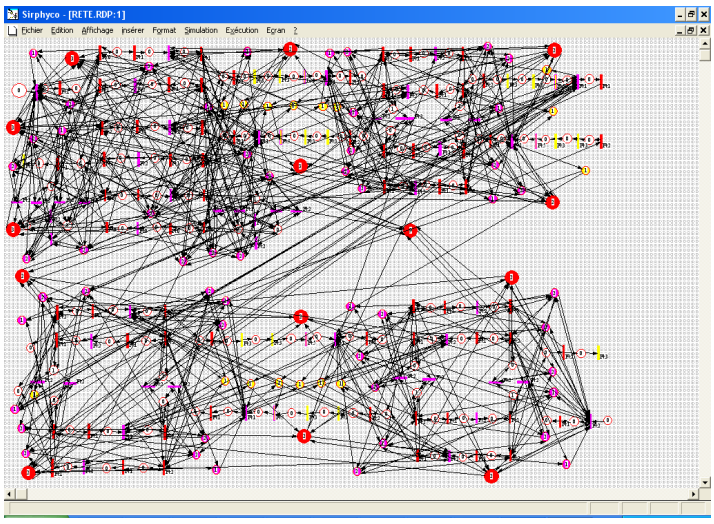


Figura 5.1: Interfaccia grafica di SIRPHYCO.

Perchè abbiamo deciso di creare HYPENS?

1. Per creare un simulatore scritto interamente in Matlab, un linguaggio di alto livello che si è imposto in ambito ingegneristico mondiale come efficace strumento di calcolo ed elaborazione;

2. Per creare un simulatore di facile utilizzo, anche a livello didattico;

3. Per creare un simulatore in grado di simulare indifferentemente reti di Petri temporizzate, reti di Petri continue e reti di Petri ibride;

4. Per creare un simulatore in grado di gestire i conflitti tra le transizioni e in grado di far evolvere qualsiasi rete.

5. Per far evolvere la rete in base ad una funzione obiettivo, se si trattano reti di Petri continue o ibride.

6. Per creare uno strumento in grado di fornire dati statistici sia numericamente che graficamente;

5.4 Funzione enter_HPN.m

Il file *enter_HPN* è stato realizzato attraverso la seguente $function$:

```
[Pre, Post, M0, vel, v, D, s, alfa] = enter_HPN
```

In ingresso non dobbiamo inserire alcun parametro in quanto la rete viene creata grazie ad una procedura guidata che chiede all'utente di inserire i dati volta per volta e, una volta inseriti, di premere il tasto $invio$; in uscita vengono restituiti tutti i parametri che individuano univocamente la rete i quali saranno l'input per il file *simulator_HPN.m*. Andiamo ad analizzare nel dettaglio i dati che ci vengono chiesti, a partire dal momento in cui mandiamo in esecuzione il file, e i dati che ci vengono restituiti in uscita .

5.4.1 Dati in ingresso

Nel momento in cui mandiamo in esecuzione il file $enter\_HPN.m$ ci appare a video la seguente richiesta:

- **Selezionare la lingua: 1(italiano) / 0(inglese)**
  Please, select your language: 1(italian) / 0(english)

Andiamo a decidere la lingua che vogliamo utilizzare durante tutta l'esecuzione del file $enter\_HPN.m$. Una volta che settiamo il parametro associato alla lingua,

inizia la fase di creazione della rete; bisogna sottolineare che se si fornisce un valore diverso da 0 e 1, viene settata automaticamente la lingua inglese.

Supponiamo di aver settato il valore 1 cioè di aver deciso di visualizzare tutte le richieste (con gli eventuali messaggi di errore) in italiano; d'ora in avanti entriamo nella fase di creazione vera e propria della rete che consiste nell'inserire i dati che ci vengono richiesti e memorizzarli in delle variabili interne alla $function$ premendo il tasto $invio$. Nell'ordine vengono visualizzate a video le **seguenti richieste**:

- **Inserire il numero di posti continui presenti nella rete:**
  Se nella rete ci sono posti continui, inseriamo il valore che ne indica il numero altrimenti settiamo il valore nullo.

- **Inserire il numero di posti discreti presenti nella rete:**
  Se nella rete ci sono posti discreti, inseriamo il valore che ne indica il numero altrimenti settiamo il valore nullo.

- **Inserire il numero di transizioni continue presenti nella rete:**
  Se nella rete ci sono transizioni continue, inseriamo il valore che ne indica il numero altrimenti settiamo il valore nullo.

- **Inserire il numero di transizioni discrete presenti nella rete:**
  Se nella rete ci sono transizioni discrete, inseriamo il valore che ne indica il numero altrimenti settiamo il valore nullo.

- **Digitare la matrice di Pre_incidenza_CC della rete che si vuole considerare:** E' una matrice di $Pre\_incidenza$ che riguarda solo la parte continua, che tiene conto di tutti gli archi Pre che collegano un posto continuo P_c con una transzione continua T_c; se nella rete non ci sono suddetti collegamenti tale richiesta non è visualizzata.

- **Digitare la matrice di Pre_incidenza_CD della rete che si vuole considerare:** E' una matrice di $Pre\_incidenza$ che riguarda la parte ibrida, che tiene conto di tutti gli archi Pre che collegano un posto continuo P_c con una transzione continua T_d; se nella rete non ci sono suddetti collegamenti tale richiesta non è visualizzata.

- **Digitare la matrice di Pre_incidenza_DC della rete che si vuole considerare:** E' una matrice di $Pre\_incidenza$ che riguarda la parte ibrida, che tiene conto di tutti gli archi Pre che collegano un posto continuo P_d con una transzione continua T_c; se nella rete non ci sono suddetti collegamenti tale richiesta non è visualizzata.

- **Digitare la matrice di Pre_incidenza_DD della rete che si vuole considerare:** E' una matrice di $Pre\_incidenza$ che riguarda la parte discreta, che tiene conto di tutti gli archi Pre che collegano un posto continuo P_d con una transzione continua T_d; se nella rete non ci sono suddetti collegamenti tale richiesta non è visualizzata.

- **Digitare la matrice di Post_incidenza_CC della rete che si vuole considerare:** E' una matrice di $Post\_incidenza$ che riguarda solo la parte continua, che tiene conto di tutti gli archi $Post$ che collegano una transizione continua T_c con un posto continuo P_c; se nella rete non ci sono suddetti collegamenti tale richiesta non è visualizzata.

- **Digitare la matrice di Post_incidenza_CD della rete che si vuole considerare:** E' una matrice di $Post\_incidenza$ che riguarda solo la parte ibrida, che tiene conto di tutti gli archi $Post$ che collegano una transizione discreta T_d con un posto continuo P_c; se nella rete non ci sono suddetti collegamenti tale richiesta non è visualizzata.

- **Digitare la matrice di Post_incidenza_DD della rete che si vuole considerare:** E' una matrice di $Post\_incidenza$ che riguarda solo la parte discreta, che tiene conto di tutti gli archi $Post$ che collegano una transizione discreta T_d con un posto discreto P_d; se nella rete non ci sono suddetti collegamenti tale richiesta non è visualizzata.

Si noti che la matrice $Post\_incidenza\_DC$ non è stata richiesta in quanto deve essere uguale alla matrice $Pre\_incidenza\_DC$ per costruzione; inoltre, qualora vengano inseriti dei valori errati, quali ad esempio dimensionamento errato delle matrici, viene visualizzato a schermo un messaggio di errore con il relativo errore commesso e con la relativa possibilità di correzione.

Una volta terminata la parte di creazione della rete, si devono assegnare tutti i rispettivi parametri che la contraddistinguono; di seguito vengono elencati le altre richieste a schermo che l'utente visualizza:

- **Inserire la marcatura iniziale dei posti P_c in un vettore riga tipo** $[M(P_1), \cdots, M(P_c)]$**:** Dobbiamo inserire un vettore riga (che verrà memorizzato come vettore colona) indicante il numero di gettoni presente nei posti continui P_c; se nella rete non ci sono posti P_c, tale richiesta non è visualizzata.

- **Inserire la marcatura iniziale dei posti P_d in un vettore riga tipo** $[M(P_c+1), \cdots, M(P)]$**:** Dobbiamo inserire un vettore riga (che verrà memorizzato come vettore colona) indicante il numero di gettoni presente nei posti continui P_d; se nella rete non ci sono posti P_d, tale richiesta non è visualizzata.

D'ora in poi vengono richiesti parametri che riguardano solo la parte discreta che riguardano solamente le transizioni T_d; quindi, se stiamo analizzando una rete in cui non ho transizioni discrete, tutto ciò che segue non verrà visualizzato.

- **Le transizioni discrete possono avere un tempo di evoluzione deterministico oppure random; di seguito sono elencati sia il tipo di evoluzione che il rispettivo numero che lo richiama: Deterministic(1), Exp(2), Unif(3), Poiss(4), Rayl(5), Wbl(6). Digitare a per visualizzare ulteriori funzioni altrimenti premere un qualsiasi tasto**.
  In questo passo viene scelto il tipo di transizioni discrete all'interno della rete; se si premere a vengono visualizzate fino a un totale di 19 distribuzioni; una volta che si è scelto il tipo di transizione bisogna immettere il/i relativo/i parametro/i che lo caratterizza.

- **Inserire il vettore riga $[s(T_c+1),\cdots,s(T)]$ indicante la transizione discreta con la rispettiva semantica di servente:0(semantica di servente infinita) oppure n(semantica di servente ennesima)**.
  Viene richiesto il numero di serventi associato ad ogni transizione discreta.

- **-Scatto in base al peso associato a ogni transizione (premi 1)**
  -Scatto in base all'indice associato a ogni transizione (premi 2)
  -Scatto in base a dei livelli di priorità e peso associato a ogni transizione premi 3). Si sceglie la politica di scatto dele transizioni in caso di conflitto generale (definizione 2.2.8) se si sceglie $1)$ o $2)$ (tecnica di risoluzione secondo l'algoritmo 2.2.2) bisogna introdurre solo un vettore riga indicante la rispettiva politica di scatto scelta, mentre nel caso $3)$ (tecnica di risoluzione secondo l'algoritmo 2.2.3) bisogna introdurre una ulteriore riga che indica il livello di priorità associato alle transizioni.

5.4.2 Dati restituiti in uscita

I dati che vengono restituiti dal file $enter\_HPN.m$ sono la base per una corretta simulazione da parte del file $simulator\_HPN.m$ e sono strutturati in modo tale che lo stesso file $simulator\_HPN.m$ possa interpretare in modo giusto la rete; quindi in uscita non avremo nient'altro che i dati forniti in ingresso, scritti secondo una forma più compatta dopo essere stati controllati. Analizziamo a blocchi l'output della $function$ in questione:

- **Matrici Pre e Post**

$$Pre = \begin{bmatrix} Pre\_CC & NaN & Pre\_CD \\ NaN & NaN & NaN \\ Pre\_DC & NaN & Pre\_DD \end{bmatrix}$$

$$Post = \begin{bmatrix} Post\_CC & NaN & Post\_CD \\ NaN & NaN & NaN \\ Post\_DC & NaN & Post\_DD \end{bmatrix}$$

Queste matrici che ci vengono restituite sono le matrici Pre e $Post$ dell'intera rete: per una questione di leggibilità, si è deciso di memorizzare i vari blocchi, che abbiamo precendentemente fornito in ingresso, attravarso una riga e una colonna di NaN (not-a-number).

- **Marcatura iniziale**

$$M0 = \begin{bmatrix} M(P_1) \\ M(P_2) \\ \vdots \\ M(P) \end{bmatrix}$$

Questo è il vettore colonna indicante la marcatura iniziale al tempo $\tau = 0$; è costituito dalla concatenazione della marcatura continua con la marcatura discreta. Come si può notare, nonostante abbiamo inserito in ingresso dei vettori riga per indicare la marcatura della rete (per una questione di leggibilità maggiore a schermo), in uscita sono stati memorizzati in un unico vettore colonna per analogia con la teoria delle reti di Petri. Se nella rete non ci sono posti continui, il vettore M_0 coinciderà con il vettore della sola marcatuta discreta e viceversa se non ho posti discreti nella rete.

- **Velocità delle transizioni continue**

$$vel = \begin{bmatrix} vel\_min(T_1) & vel\_max(T_1) \\ \vdots & \vdots \\ vel\_min(T_c) & vel\_max(T_c) \end{bmatrix}$$

Questa matrice mi indica, nel caso ci siano transizioni continue nella rete, la velocità minima e massima che ciascuna può assumere; se $T_c = 0$ allora la matrice vel sarà una matrice vuota.

- **Parametri per il calcolo dei timer delle transizioni** T_d

$$D = \begin{bmatrix} param\_1(T_c+1) & param\_2(T_c+1) & param\_3(T_c+1) \\ \vdots & \vdots & \vdots \\ param\_1(T) & param\_2(T) & param\_3(T) \end{bmatrix}$$

La matrice D è associata alle transizioni discrete e fornisce il relativo parametro (o i relativi parametri) per calcolare i tempi di scatto delle transizioni discrete temporizzate. Se $T_d = 0$ allora D ha dimensione nulla.

- **Tipo di transizione discreta, serventi per ogni transizione e grado di priorità associato**

$$v = \begin{bmatrix} v(T_c+1) & \cdots & v(T) \end{bmatrix}$$
$$s = \begin{bmatrix} s(T_c+1) & \cdots & s(T) \end{bmatrix}$$
$$alfa = \begin{bmatrix} alfa(1,T_c+1) & \cdots & alfa(1,T) \\ alfa(2,T_c+1) & \cdots & alfa(2,T) \end{bmatrix}$$

Il vettore v indica il tipo di transizione discreta temporizzata, cioè se essa è deterministica o stocastica e, in quest'ultimo caso, che distribuzione assume. Il vettore s è quello che tiene conto della politica dei serventi associati a ciascuna transizione discreta mentre $alfa$ può essere o un vettore riga, se si scegli di risolvere i conflitti generali senza la considerazione dei livelli di priorità, oppure una matrice nel caso contrario dove la seconda colonna sta ad indicare il livello di appartenenza delle transizioni. Il vettore e, il vettore s e la matrice (o vettore) $alfa$ esistono se nella rete da simulare ci sono transizioni discrete altrimenti vengono posti tutti con dimensione nulla.

5.5 Funzione make_HPN.m

Il file *make_HPN* è stato realizzato attraverso la seguente $function$:

```
[Pre,Post,M0,vel,v,D,s,alfa]=make_HPN(Pre_CC,Pre_CD,
Pre_DC,Pre_DD,Post_CC,Post_CD,Post_DC,Post_DD,M0_Pc,
M0_Pd,vel,v,D,s,alfa,it)
```

In ingresso dobbiamo inserire i parametri della rete i quali vengono analizzati; se non ci sono errori di dimensionamento fra le varie matrici o incompatibilità di dati inseriti, in uscita vengono restituiti i parametri che individuano univocamente la rete i quali saranno l'input per il file *simulator_HPN.m*. Andiamo ad analizzare nel dettaglio i dati da inserire in ingresso e i dati che ci vengono restituiti in uscita.

5.5.1 Dati in ingresso

Pre_CC**:** Matrice degli archi che uniscono un posto continuo P_c ad una transizione continua T_c la cui dimensione è $m_c \times n_c$; se nella rete $P_c = 0$ oppure $T_c = 0$ allora la matrice Pre_CC è vuota.

Pre_CD**:** Matrice degli archi che uniscono un posto continuo P_c ad una transizione continua T_d la cui dimensione è $m_c \times n_d$; se nella rete $P_c = 0$ oppure $T_d = 0$ allora la matrice Pre_CD è vuota.

Pre_DC**:** Matrice degli archi che uniscono un posto continuo P_d ad una transizione continua T_c la cui dimensione è $m_d \times n_c$; se nella rete $P_d = 0$ oppure $T_c = 0$ allora la matrice Pre_DC è vuota.

Pre_DD**:** Matrice degli archi che uniscono un posto continuo P_d ad una transizione continua T_d la cui dimensione è $m_d \times n_d$; se nella rete $P_d = 0$ oppure $T_d = 0$ allora la matrice Pre_DD è vuota.

$Post\_CC$**:** Matrice degli archi che uniscono una transizione continua T_c ad un posto continuo P_c la cui dimensione è $m_c \times n_c$; se nella rete $P_c = 0$ oppure $T_c = 0$ allora la matrice $Post\_CC$ è vuota.

$Post\_CD$**:** Matrice degli archi che uniscono una transizione continua T_d ad un posto continuo P_c la cui dimensione è $m_c \times n_d$; se nella rete $P_c = 0$ oppure $Tdc = 0$ allora la matrice $Post\_CD$ è vuota.

$Post\_DC$**:** Matrice degli archi che uniscono una transizione continua T_c ad un posto continuo P_d la cui dimensione è $m_d \times n_c$; se nella rete $P_d = 0$ oppure $T_c = 0$ allora la matrice $Post\_DC$ è vuota. Per costruzione tale matrice deve essere uguale a Pre_DC.

$Post\_DD$**:** Matrice degli archi che uniscono una transizione continua T_d ad un posto continuo P_d la cui dimensione è $m_d \times n_d$; se nella rete $P_d = 0$ oppure $T_d = 0$ allora la matrice $Post\_DD$ è vuota.

$M0\_P_c$**:** Valore della marcatura iniziale dei posti continui P_c che indica il numero di gettoni contenuti al tempo iniziale $\tau = 0$; la dimensione del vettore che inseriamo è $1 \times m_c$ (per questioni di leggibilità) ma verrà memorizzato come un vettore colonna per uniformità con la teoria; se nella rete $P_c = 0$ allora il vettore $M0\_P_c$ è vuoto.

$M0\_P_d$**:** Valore della marcatura iniziale dei posti discreti P_d che indica il numero di gettoni contenuti al tempo iniziale $\tau = 0$; la dimensione del vettore che inseriamo è $1 \times m_d$ (per questioni di leggibilità) ma verrà memorizzato come

un vettore colonna per uniformità con la teoria; se nella rete $P_d = 0$ allora il vettore $M0\_P_d$ è vuoto.

vel**:** Matrice delle velocità vel che associa rispettivamente, a ogni transizione continua T_c (righe della matrice), un valore minimo e massimo di velocità (colonne della matrice); la dimensione della matrice è $n_c \times 2$. Se nella rete $T_c = 0$ allora la matrice vel è vuota.

v**:** Vettore che associa il tipo di transizione (deterministica, esponenziale, uniforme, etc) a ciascuna transizione discreta T_d; la dimensione del vettore è $1 \times n$ e le prime colonne, fino a T_c avranno valore pari a NaN. Se nella rete $T_d = 0$ allora il vettore v è vuoto.

D**:** Matrice contenente i valori utili a calcolare i tempi associati a ogni transizione T_d; è fortemente legata al tipo di transizione indicato dal vettore v il quale, a seconda del valore memorizzato per ogni colonna, segnala al simulatore il numero di parametri che deve andare a leggere per ogni riga di D. la dimensione della matrice è $n \times 3$ e e le prime colonne, fino a T_c avranno valore pari a NaN. Se nella rete $T_d = 0$ allora la matrice D è vuota.

s**:** Vettore che associa a ogni transizione T_d la politica di servente che può essere k-finita(valore k) o infinita (valore 0); la dimensione del vettore è $1 \times n$ e le prime colonne, fino a T_c avranno valore pari a NaN. Se nella rete $T_d = 0$ allora il vettore s è vuoto.

$alfa$**:** Può essere definito come un vettore oppure come una matrice e ha importanza nel momento in cui ci troviamo di fronte a un conflitto generale (paragrafo 2.2.8); se scriviamo $alfa$ come un vettore possiamo inserire ho un vettore con tutti i valori diversi da 0 (nel qual caso stiamo attribuendo un peso alla transizioni e stiamo imponendo che scatterà prima la transizione con peso maggiore) oppure tutti nulli (in questo caso scatterà prima la transizione con indice minore). Se scriviamo, invece, $alfa$ come una matrice, per la prima riga continua a valere quanto detto sopra, mentre nella seconda riga stiamo associando alle transizioni un livelo di priorità (livello più basso $\rightarrow$ priorità maggiore). Il numero di righe può unitario o doppio, il numero di colonne è pari a n_d dove le prime colonne fino a T_c avranno valore NaN.

it**:** Parametro che setta la lingua italiana se $it = 1$ oppure la lingua inglese $it = 0$.

5.5.2 Dati restituiti in uscita

I dati che vengono restituiti in uscita sono analoghi a quelli della funzione $enter\_HPN$; pertanto si veda il paragrafo 5.4.2.

5.6 Funzione simulator_HPN.m

Il file *make_HPN* è stato realizzato attraverso una $function$:

```
[Type,M_Evol,IFS_Evol,P_macro,Event_macro_Evol,
firing_transition,timer_macro_event,tau,Q_Evol,
Pc_Pd_Tc_Td]=simulator_HPN(Pre,Post,M,vel,v,D,s,
alfa,time_stop,simulation_type,it)
```

Si può notare, a prima vista, che in ingresso gli devo passare l'uscita di uno dei 2 file precedentemente trattati e cioè di $enter\_HPN.m$ oppure $make\_HPN.m$: questo era un aspetto che ci si poteva facilmente aspettare in quanto il file $simulator\_HPN$ non fa altro che simulare la rete che abbiamo creato precedentemente con uno dei 2 file di interfaccia.

Analizziamo nel dettaglio i dati che dobbiamo inserire in ingresso e i dati che ci vengono restituiti in uscita .

5.6.1 Dati in ingresso

- **Uscita di** $enter\_HPN$ **o** $make\_HPN$

  $$Pre, Post, M, vel, v, D, s, alfa$$

  Sono le matrici di Pre incidenza, $Post$ incidenza, la marcatura iniziale M, la matrice delle velocità vel, il vettore v indicante i tipi di transizione discreta, la matrice D con i parametri per il calcolo dei clock delle transizioni discrete, il vettore dei serventi s e il vettore (o matrice) $alfa$ per la risoluzione dei conflitti generali.

- **Durata della silulazione**

  $$time\_stop$$

  Valore che indica per quanto tempo si vuole far evolvere la rete ponendo un limite alla simulazione.

- **Scelta del tipo di simulazione**

  $$simulation\_type$$

  Il settaggio di questa variabile permette di decidere il tipo di simulazione in base alle esigenze dell'utente:

1. $simulation\_type = 2$ se si vuole una simulazione step-by-step tramite la quale viene visualizzata a schermo l'evoluzione della rete per ogni scatto delle transizioni (Marcatura corrente, vettore di abilitazione, eventuali velocità e eventuali orologi associati alle transizioni); l'utente guida l'evoluzione stessa della rete premendo il tasto invio dopo lo scatto di ogni transizione fino a un tempo pari a $time\_stop$.
2. $simulation\_type = 1$ se si vuole una simulazione senza interruzione visualizzando a schermo l'evoluzione della rete per ogni scatto delle transizioni (Marcatura corrente, vettore di abilitazione, eventuali velocità e eventuali orologi associati alle transizioni); il comportamento si differenzia dal punto precedente per il fatto che ora non è più l'utente a guidare l'evoluzione della rete (premendo il tasto invio) ma la rete stessa evolverà da sola fino a un tempo pari a $time\_stop$
3. $simulation\_type = 0$ se non si vuole visualizzare nulla a schermo poichè si è interessati solamente ai dati finali da analizzare con il file successivo $analysis\_HPN.m$

- **Scelta della lingua**

  $$it$$

  Parametro che setta la lingua italiana se $it = 1$ oppure la lingua inglese $it = 0$

5.6.2 Dati restituiti in uscita

- **Tipo di rete**

  $$Type$$

  Identifica se la rete di Petri è continua $Type = 1$, discreta $Type = 2$ oppure ibrida $Type = 3$.

- **Evoluzione della marcatura**

  $$M\_Evol$$

  Memorizza i valori che la marcatura ha avuto durante l'evoluzione fino al tempo $time\_stop$.

- **Evoluzione delle velocità**

  $$IFS\_Evol$$

  Memorizza i valori che le velocità, associate alle transizioni continue, hanno assunto in relazione alla funzione obiettivo durante l'evoluzione fino al tempo $time\_stop$; se $T_c = 0$ allora IFS_Evol sarà un vettore di dimensioni nulle.

- **Posto che genera un macroevento e il relativo macroevento**

$$P\_macro \quad Event\_macro\_Evol$$

Memorizzano rispettivamente i valori dei posti continui che generano macroeventi durante l'evoluzione fino al tempo $time\_stop$ e il relativo tipo di macroevento:

1. $Event\_macro\_Evol = 0$ vuol dire che il posto $P = P\_macro$ si è svuotato;
2. $Event\_macro\_Evol = 1$ vuol dire che il posto $P = P\_macro$ abilita una transizione discreta al tempo τ;
3. $Event\_macro\_Evol = -1$ vuol dire che il posto $P = P\_macro$ disabilita una transizione discreta al tempo τ;

Se $P_c = 0$ sia P_macro che $Event\_macro\_Evol$ saranno dei vettori vuoti mentre se $P_c > 0$ e il macroevento è generato dallo scatto di una transizione discreta (e non a causa di un posto continuo), P_macro ed $Event\_macro\_Evol$ avranno dei valori pari a NaN.

- **Transizioni discrete scattanti**

$$firing\_transition$$

Memorizza le transizioni discrete scattate durante l'evoluzione fino al tempo $time\_stop$. Se $T_d = 0$, $firing\_transition$ sarà un vettore vuoti mentre se $T_d > 0$ e il macroevento è generato in seguito all'evoluzione di un posto continuo, $firing\_transition$ avrà un valore pari a NaN.

- **Tempo di macroevento**

$$timer\_macro\_event$$

Identifica il tempo associato al macroevento che si è verificato; se non si verifica nessun macroevento allora $timer\_macro\_event$ avrà un valore pari a NaN.

- **tempo totale**

$$tau$$

Identifica il tempo di simulazione totale a partire dall'istante 0.

- **Evoluzione della matrice degli orologi**

  $$Q\_Evol$$

  Memorizza l'evoluzione nel tempo della matrice degli orologi associati a ogni transizione T_d per ogni marcatura M raggiunta.

- **Posti e transizioni nella rete**

  $$P_c\_P_d\_T_c\_T_d$$

  E' un vettore che memorizza i valori di P_c, P_d, T_c e T_d.

5.7 Funzione analysis_HPN.m

Il file *analysis_HPN* è stato realizzato attraverso una $function$:

```
[M_medium,M_max, V_medium, T_medium, Pd_probability,
Marking_Asymptote]=analysis_HPN(Evol,statistic_plot,
graph,marking_plot,Transition_firing_plot,up_marking
_plot, Pd_probability_plot, Marking_mean_Asymptote,
mean_speed_plot,Transition_frequency_plot,it)
```

Esso ci permette di effettuare delle statistiche sui risultati ottenuti in seguito alla simulazione fatta con il file $simulator\_HPN$, la cui uscita viene richiamata in ingresso nel file $analysis\_HPN$ tramite l'array di celle $Evol$; quindi, affinchè funzioni il file $analysis\_HPN$, siamo costretti a memorizzare l'uscita del file $simulator\_HPN$ all'interno di un array di celle: questo permette di ridurre notevolmente il numero di parametri da dare in ingresso al file $analysis\_HPN$.

5.7.1 Dati in ingresso

- **Uscita del file** $simulator\_HPN$

  $$Evol$$

  E' un array di celle costituito dalla concatenazione di più celle: ogni cella contiene un parametro di uscita del file $simulator\_HPN$.

- **Visualizzazione grafica delle statistiche**

  $$statistic\_plot$$

Se pongo $statistic\_plot = 1$, vengono visualizzati 2 istogrammi in 2 finestre diverse: uno visualizza in ascissa i posti e in ordinata il valore massimo della marcatura raggiunta al tempo $\tau = time\_stop$; l'altro visualizza in ascissa i posti e in ordinata il valore medio della marcatura raggiunta al tempo $\tau = time\_stop$.

- **Grafo di evoluzione**

$$graph$$

E' un parametro che può essere utilizzato solo si si sta simulando una rete di Petri discreta: se metto $graph = 1$, viene visualizzato il grafo di evoluzione della TPN, dove a ogni stato è associata la marcatura corrente M, la matrice degli orologi Q, le transizioni scattanti con il relativo tempo di scatto e il tempo totale di simulazione τ.

- **Grafico delle marcature in diverse finestre**

$$marking\_plot$$

E' un parametro che può essere un numero o un array e serve per visualizzare graficamente, in finestre diverse, l'evoluzione della marcatura nei posti fino al tempo $\tau = time\_stop$. Nel grafico rappresentiamo il tempo nell'asse delle ascisse e il valore della marcatura nell'asse delle ordinate. Si può decidere se fare il grafico solo di alcuni posti, mettendo ad esempio $marking\_plot = [x, y, z]$ per avere i grafici della marcatura dei posti p_x, p_x e p_z, oppure di fare il grafico di tutte le marcature dei posti ponendo $marking\_plot = -1$; in quest'ultimo caso si poteva anche creare un vettore $marking\_plot$ contenente gli indici di tutti i posti della rete, ma per semplicità e leggibilità si è deciso di ottenere lo stesso risultato imponendo $marking\_plot = -1$.

- **Grafico degli scatti delle transizioni** Se pongo $Transition\_firing\_plot = 1$, vengono visualizzate,in un unico grafico, gli scatti delle transizioni nei rispettivi istanti di tempo in cui scattano, fino al tempo massimo $\tau = time\_stop$. Nel grafico rappresentiamo il tempo nell'asse delle ascisse e le transizioni discrete nell'asse delle ordinate.

- **Grafico delle marcature in un'unica finestra**

$$up\_marking\_plot$$

E' un parametro che può essere un numero o un array e serve per visualizzare graficamente, nella stessa finestra, la sovrapposizione dell'evoluzione

della marcatura nei posti fino al tempo $\tau = time\_stop$. Nel grafico rappresentiamo il tempo nell'asse delle ascisse e il valore delle marcatura nell'asse delle ordinate differenziate, a seconda del posto, da un colore diverso riconducibile al posto stesso grazie alla presenza di una finestra che mi indica la legenda. Si può decidere se fare il grafico della sovrapposizione solo di alcuni posti, mettendo ad esempio $up\_marking\_plot = [x, y, z]$ per avere il grafico delle marcature sovrapposte dei posti p_x, p_x e p_z, oppure di fare il grafico della sovrapposizione di tutte le marcature nei posti ponendo $up\_marking\_plot = -1$; in quest'ultimo caso si poteva anche creare un vettore $up\_marking\_plot$ contenente gli indici di tutti i posti della rete, ma per semplicità e leggibilità si è deciso di ottenere lo stesso risultato imponendo $up\_marking\_plot = -1$.

- **Grafico delle probabilità di avere x gettoni nei posti**

$$P_d\_probability\_plot$$

Se pongo $P_d\_probability\_plot = 1$, vengono visualizzate, in diverse finestre tante quanti sono i posti discreti p_d, le probabilità di avere x gettoni nel posto $p_i \in p_d$ fino al tempo massimo $\tau = time\_stop$. Nel grafico rappresento il numero di gettoni nell'asse delle ascisse e la probabilità corrispondente al numero di gettoni nell'asse delle ordinate.

- **Grafico del valor medio della marcatura nei posti**

$$Marking\_mean\_Asymptote$$

E' un parametro che può essere un numero o un array e serve per visualizzare graficamente, in finestre diverse, l'evoluzione del valor medio dei gettoni nei posti fino al tempo $\tau = time\_stop$. Nel grafico rappresentiamo il tempo nell'asse delle ascisse e il valor medio della marcatura di un posto nell'asse delle ordinate. Dato $x(p_i, j) = Marking\_Asymptote(p_i, j)$, cioè il valore medio della marcatura del posto p_i all'istante τ_j in cui si è verificato l'evento $\bar{\eta_j}$, l'evoluzione nel tempo del valor medio della marcatura del posto p_i la calcoliamo come

$$x(p_i, j) = \frac{(x(i, j-1) * \tau(j) + (\tau(j+1) - \tau(j)) * M(i, y(j)))}{\tau(j+1)}$$

$$\forall j = \Delta(k), k = 0, 1, 2, \cdots$$

Si può decidere se fare il grafico solo di alcuni posti, mettendo ad esempio $Marking\_mean\_Asymptote = [x, y, z]$ per avere i grafici dell'evoluzione

del valor medio della marcatura nei posti p_x, p_x e p_z, oppure di fare il grafico dell'evoluzione del valor medio della marcatura in tutti i posti ponendo $Marking\_mean\_Asymptote = -1$; in quest'ultimo caso si poteva anche creare un vettore $Marking\_mean\_Asymptote$ contenente gli indici di tutti i posti della rete, ma per semplicità e leggibilità si è deciso di ottenere lo stesso risultato imponendo $Marking\_mean\_Asymptote = -1$.

- **Grafico della velocità media delle transizioni continue**

$$mean\_speed\_plot$$

Se pongo $mean\_speed\_plot = 1$, viene visualizzato,in un unico grafico, il valor medio delle transizioni continue fino al tempo massimo $\tau = time\_stop$. Nel grafico rappresentiamo le transizioni continue nell'asse delle ascisse e il valor medio della velocità associato alle transizioni stesse nell'asse delle ordinate

- **Grafico della frequenza di scatto delle transizioni**

$$Transition\_frequency\_plot$$

Se pongo $Transition\_frequency\_plot = 1$, viene visualizzato,in un unico grafico, la frequenza di scatto delle transizioni discrete fino al tempo massimo $\tau = time\_stop$. Nel grafico rappresentiamo le transizioni discrete nell'asse delle ascisse e il la frequenza di scatto delle transizioni stesse nell'asse delle ordinate.

- **Scelta della lingua**

$$it$$

Se pongo $it = 0$ imposto la lingua inglese, se metto $it = 1$ imposto la lingua italiana. Di default è impostato $it = 1$.

Capitolo 6

Esempi di simulazione e analisi

Si utilizzeranno 3 delle 4 $function$ create:

1. *make_HPN.m* (interfaccia)

2. *simulator_HPN.m* (simulatore)

3. *analysis_HPN.m* (analisi della simulazione)

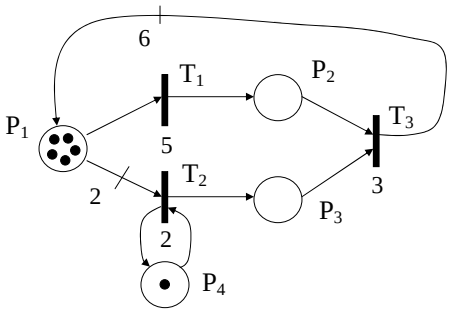
Si è tralasciato l'utilizzo del file $enter\_HPN.m$ in quanto otteniamo lo stesso risultato di $make\_HPN.m$ ma con un procedimento meno rapido.

6.1 Simulazione di reti di Petri discrete temporizzate

6.1.1 TPN con transizioni a ritardo deterministico

Esempio 6.1.1. *Si vuole simulare per un tempo pari a* $\tau = 12$ *la rete in Figura 6.1*

1. *Creiamo un file* $demo.m$ *all'interno del quale memorizziamo i parametri*



Livello di priorità		Peso
T_2	level 2	$\alpha_1 = \alpha_2 = \alpha_3 = 1$
T_1 , T_3	level 1	

Figura 6.1: Rapprenentazione grafica di una TPN

della rete e mandiamolo in esecuzione :

$$Pre\_CC = [\,]; \qquad Pre\_CD = [\,];$$

$$Pre\_DC = [\,]; \qquad Pre\_DD = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix};$$

$$Post\_CC = [\,]; \qquad Post\_CD = [\,];$$

$$Post\_DC = [\,]; \qquad Post\_DD = \begin{bmatrix} 0 & 0 & 6 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix};$$

$$M0\_P_c = [\,]; \qquad M0\_P_d = [5\ 0\ 0];$$

$$vel = [\,]; \qquad v = [1\ 1\ 1];$$

$$D = \begin{bmatrix} 5 & NaN & NaN \\ 2 & NaN & NaN \\ 3 & NaN & NaN \end{bmatrix}; \qquad s = [0\ 1\ 0];$$

$$alfa = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix};$$

Non visualizziamo alcuna informazione a video in quanto è stato messo un ; alla fine di ogni parametro.

2. *Memorizziamo i dati relativi alla rete all'interno di un array di celle che chiamiamo per semplicità* A

$$A = (Pre\_CC, Pre\_CD, Pre\_DC, Pre\_DD, Post\_CC, Post\_CD, Post\_DC,$$

$$Post\_DD, M0\_Pc, M0\_Pd, vel, v, D, s, alfa);$$

Non visualizziamo alcuna informazione a video in quanto è stato messo un ; alla fine dell'array di celle A*.*

3. *Mandiamo in esecuzione il file* $make\_HPN.m$ *che avrà in ingresso l'array di celle* A *appena creato e poi un valore pari a* 1 *che mi indica che voglio utilizzare la lingua italiana.*

$$rete = make\_HPN(A, 1)$$

In uscita visualizziamo le seguenti informazioni

```
transizione T1 -->servente inf; peso 1;
                  livello priorità 1;
transizione T2 -->servente singolo; peso 1;
                  livello priorità 2;
transizione T3 -->servente inf; peso 1;
                  livello priorità 1;
```

dove

$$rete = \{Pre, Post, M0, vel, v, D, s, alfa\}$$

4. *Ora dobbiamo mandare in esecuzione il file* $simulator\_HPN.m$ *che avrà in ingresso l'array di celle* $rete$ *appena creato, un parametro settato a* 12 *che mi indica la durata della simulazione, e un parametro che mi indica il tipo di simulazione (nel nostro caso lo mettiamo pari a* 2 *perchè vogliamo una simulazione step-by-step)*

$$Evol = simulator\_HPN(rete, 12, 2)$$

Stiamo memorizzando tutta l'evioluzione della rete all'interno di un array di celle $Evol$*: quello che visualizziamo è il seguente risultato*

```
Valore della marcatura discreta corrente
5     0     0

Valore dei tempi residui di scatto Q (orologi)
```

```
  5     5     5     5     5
  2   NaN   NaN   NaN   NaN
NaN   NaN   NaN   NaN   NaN

Valore del vettore di abilitazione e
5     1     0

La transizione discreta che scatta al tempo 2
(tempo tot=2) è T2 (1 volta)

Valore della marcatura discreta corrente
3     0     1

Valore dei tempi residui di scatto Q (orologi)
  3     3     3
  2   NaN   NaN
NaN   NaN   NaN

Valore del vettore di abilitazione e
3     1     0

La transizione discreta che scatta al tempo 2
(tempo tot=4) è T2 (1 volta)

Valore della marcatura discreta corrente
1     0     2

Valore dei tempi residui di scatto Q (orologi)
  1
NaN
NaN

Valore del vettore di abilitazione e
1     0     0

La transizione discreta che scatta al tempo 1
(tempo tot=5) è T1 (1 volta)

Valore della marcatura discreta corrente
0     1     2
```

```
Valore dei tempi residui di scatto Q (orologi)
NaN
NaN
  3

Valore del vettore di abilitazione e
0     0     1

La transizione discreta che scatta al tempo 3
(tempo tot=8) è T3 (1 volta)

Valore della marcatura discreta corrente
6     0     1

Valore dei tempi residui di scatto Q (orologi)
  5     5     5     5     5     5
  2   NaN   NaN   NaN   NaN   NaN
NaN   NaN   NaN   NaN   NaN   NaN

Valore del vettore di abilitazione e
6     1     0

La transizione discreta che scatta al tempo 2
(tempo tot=10) è T2 (1 volta)

Valore della marcatura discreta corrente
4     0     2

Valore dei tempi residui di scatto Q (orologi)
  3   3     3     3
  2 NaN   NaN   NaN
NaN NaN   NaN   NaN

Valore del vettore di abilitazione e
4     1     0

La transizione discreta che scatta al tempo 2
(tempo tot=12) è T2 (1 volta)

Valore della marcatura discreta corrente
2     0     3
```

```
Valore dei tempi residui di scatto Q (orologi)
  1   1
  2 NaN
NaN NaN

Valore del vettore di abilitazione e
2     1     0

Evol =
Columns 1 through 5
[2] [3x7 double] [1x7 double] [1x7 double] [1x7 double]
Column 6 through 7
{1x7 cell} [1x4 double]
```

5. *Non ci resta che mandare in esecuzione il file di analisi,cioè*

$$analisi = analysis\_HPN(Evol, 1, 1, 1, 1, -1, 2, [\,], [\,], [\,])$$

Nell'ordine gli sto passando in ingresso l'array di celle $Evol$ *volendo una visualizzazione grafica delle statistiche (*1*), il grafo di raggiungibilità (*1*), il grafico dell' andamento della marcatura nel posto* P_1 (1)*, il grafico della frequenza di scatto delle transizioni discrete (*1*), la sovrapposizione dei grafici degli andamenti delle marcature in tutti i posti (*-1*) e il grafico della probabilità di avere* x *gettoni nel posto* P_2*. Otteniamo i seguenti risultati:*

```
M(Pi)
5 0 0--> OROLOGI DELLE TRANSIZIONI
         T1  5  5  5  5  5
         T2  2
         Transizioni scattanti-->T2(1 volta);
         Tempo di scatto=2
         Tempo totale=2
M(Pi)
3 0 1--> OROLOGI DELLE TRANSIZIONI
         T1  3  3  3
         T2  2
         Transizioni scattanti-->T2(1 volta);
         Tempo di scatto=2
         Tempo totale=4
```

```
M(Pi)
1 0 2--> OROLOGI DELLE TRANSIZIONI
         T1  1
         Transizioni scattanti-->T1(1 volta);
         Tempo di scatto=1
         Tempo totale=5
M(Pi)
0 1 2--> OROLOGI DELLE TRANSIZIONI
         T3  3
         Transizioni scattanti-->T3(1 volta);
         Tempo di scatto=3
         Tempo totale=8
M(Pi)
6 0 1--> OROLOGI DELLE TRANSIZIONI
         T1  5  5  5  5  5  5
         T2  2
         Transizioni scattanti-->T2(1 volta);
         Tempo di scatto=2
         Tempo totale=10
M(Pi)
4 0 2--> OROLOGI DELLE TRANSIZIONI
         T1  3  3  3  3
         T2  2
         Transizioni scattanti-->T2(1 volta);
         Tempo di scatto=2
         Tempo totale=12
M(Pi)
2 0 3--> OROLOGI DELLE TRANSIZIONI
         T1  1  1
         T2  2
Fine della simulazione

E[M(P1)]=3.750
E[M(P2)]=0.250
E[M(P3)]=1.000
max(P1)=6.000
max(P2)=1.000
max(P3)=3.000
Prob[P1 = 0]=0.250
Prob[P1 = 1]=0.083
Prob[P1 = 5]=0.333
```

```
Prob[P1 = 6]=0.333
Prob[P2 = 0]=0.750
Prob[P2 = 1]=0.250
Prob[P3 = 0]=0.333
Prob[P3 = 1]=0.333
Prob[P3 = 2]=0.333
E[T1]=0.083
E[T2]=0.333
E[T3]=0.083

analisi =

[1x3 double][1x3 double][][1x3 double][3x7 double]
```

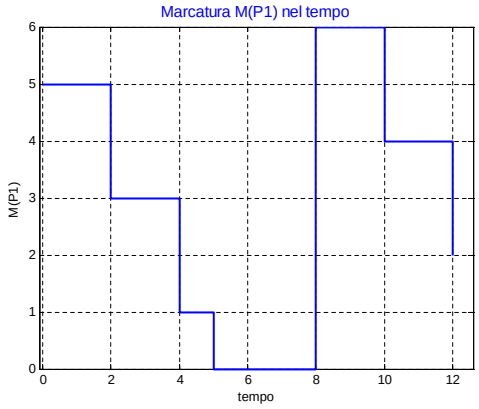


Figura 6.2: Variazione della marcatura nel posto p_1.

6.1.2 Rete ibrida

Esempio 6.1.2. *Si vuole simulare per un tempo pari a* $\tau = 240$ *la rete in Figura 6.7, con transizioni continue* t_1 *e* t_2 *e transizioni discrete* t_3 *e* t_4 *temporizzate con un ritardo rispettivamente di* $d_3 = 90$ *e* $d_4 = 60$.

1. *Creiamo un file* $demo1.m$ *all'interno del quale memorizziamo i parametri*

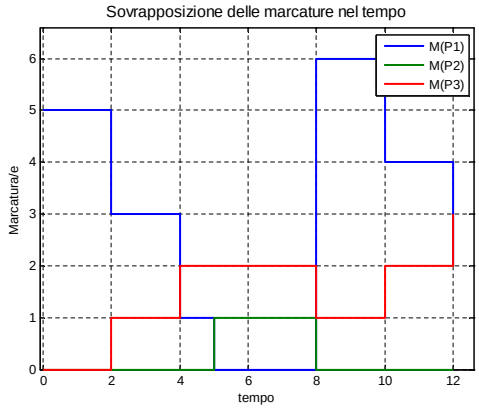


Figura 6.3: Sovrapposizione delle marcature nel tempo.

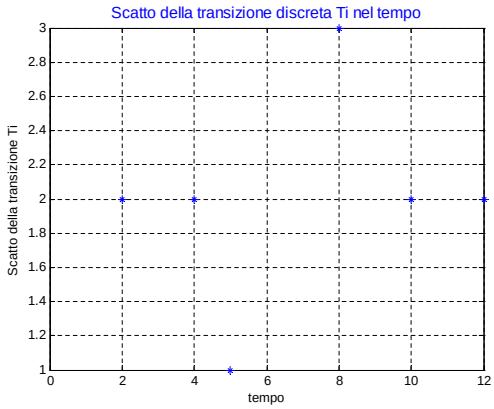


Figura 6.4: Frequenza di scatto delle transizioni.

della rete e mandiamolo in esecuzione :

$$Pre\_CC = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad Pre\_CD = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix};$$
$$Pre\_DC = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad Pre\_DD = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$Post\_CC = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad Post\_CD = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix};$$
$$Post\_DC = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad Post\_DD = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix};$$

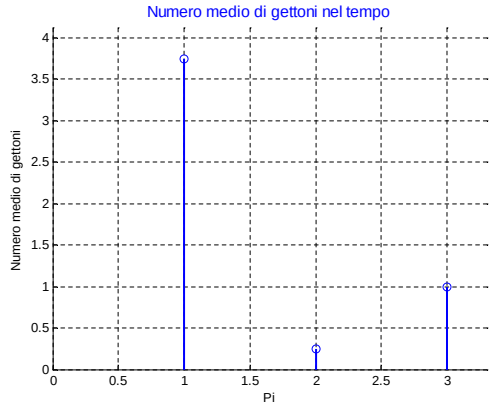


Figura 6.5: Numero medio di gettoni nei posti.

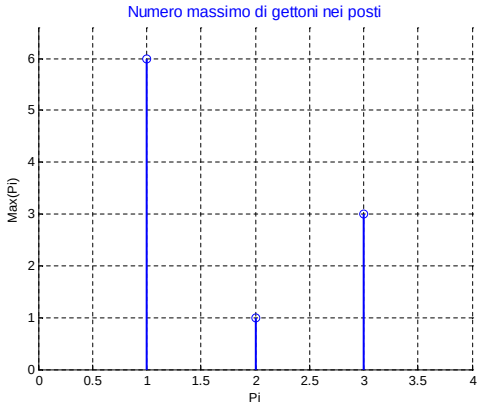


Figura 6.6: Numero massimo di gettoni nei posti.

$$M0\_P_c = [180, 0]; \qquad M0\_P_d = [1, 1];$$
$$vel = \begin{bmatrix} 0 & 3 \\ 0 & 2 \end{bmatrix}; \qquad v = [NaN\ NaN\ 1\ 1];$$
$$D = \begin{bmatrix} NaN & NaN & NaN \\ NaN & NaN & NaN \\ 90 & NaN & NaN \\ 60 & NaN & NaN \end{bmatrix}; \quad s = [NaN\ NaN\ 0\ 0];$$
$$alfa = \begin{bmatrix} NaN & NaN & 1 & 1 \end{bmatrix};$$

Non visualizziamo alcuna informazione a video in quanto è stato messo un ; alla fine di ogni parametro.

2. *Memorizziamo i dati relativi alla rete all'interno di un array di celle che*

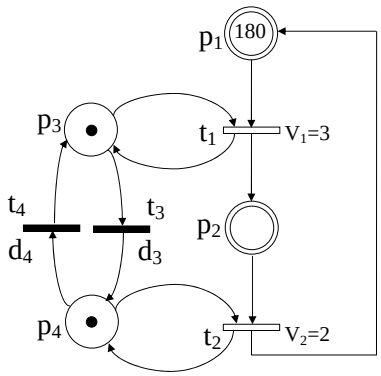


Figura 6.7: Rapprenentazione grafica di una HPN

chiamiamo per semplicità A

$$A1 = (Pre\_CC, Pre\_CD, Pre\_DC, Pre\_DD, Post\_CC, Post\_CD, Post\_DC,$$

$$Post\_DD, M0\_Pc, M0\_Pd, vel, v, D, s, alfa);$$

Non visualizziamo alcuna informazione a video in quanto è stato messo un ; alla fine dell'array di celle $A1$.

3. *Mandiamo in esecuzione il file* $make\_HPN.m$ *che avrà in ingresso l'array di celle* $A1$ *appena creato e poi un valore pari a* 1 *che mi indica che voglio utilizzare la lingua italiana.*

$$rete = make\_HPN(A1, 1)$$

In uscita visualizziamo le seguenti informazioni

```
Transizione T1 --> vel_min=0  vel_max=3
Transizione T2 --> vel_min=0  vel_max=2
transizione T3 -->servente inf; peso 1;
transizione T4 -->servente inf; peso 1;
```

dove

$$rete = \{Pre, Post, M0, vel, v, D, s, alfa\}$$

4. *Ora dobbiamo mandare in esecuzione il file* $simulator\_HPN.m$ *che avrà in ingresso l'array di celle* $rete$ *appena creato, un parametro settato a* 240 *che mi indica la durata della simulazione, un parametro che mi indica il*

tipo di simulazione (nel nostro caso lo mettiamo pari a 2 *perchè vogliamo una simulazione step-by-step), un parametro che mi indica che voglio massimizzare (*-1*) la funzione obiettivo* $J = [1\ 1]$.

$$Evol = simulator\_HPN(rete, 12, 2, -1, [11])$$

Stiamo memorizzando tutta l'evioluzione della rete all'interno di un array di celle $Evol$*: quello che visualizziamo è il seguente risultato*

```
Valore della marcatura continua corrente
180     0

vel1(fortemente abilitata)=3
vel2(non abilitata)=0

Valore della marcatura discreta corrente
1     0

Valore dei tempi residui di scatto Q (orologi)
   NaN
   NaN
    90
   NaN

Valore del vettore di abilitazione e
1     0     1     0

Si svuota il posto P1 dopo un tempo 60.000
(tempo tot=60)
Valore della marcatura continua corrente
0   180

vel1(debolmente abilitata)=0
vel2(non abilitata)=0

Valore della marcatura discreta corrente
1     0

Valore dei tempi residui di scatto Q (orologi)
   NaN
   NaN
```

```
    30
   NaN


Valore del vettore di abilitazione e
1     0     1     0

La transizione discreta che scatta al tempo 30
(tempo tot=90) è T3 (1 volta)

Valore della marcatura continua corrente
0   180

vel1(non abilitata)=0
vel2(fortemente abilitata)=2

Valore della marcatura discreta corrente
0     1

Valore dei tempi residui di scatto Q (orologi)
   NaN
   NaN
   NaN
    60

Valore del vettore di abilitazione e
0     1     0     1

La transizione discreta che scatta al tempo 60
(tempo tot=150) è T4 (1 volta)

Valore della marcatura continua corrente
120    60

vel1(fortemente abilitata)=3
vel2(non abilitata)=0

Valore della marcatura discreta corrente
1     0

Valore dei tempi residui di scatto Q (orologi)
```

```
   NaN
   NaN
    90
   NaN

Valore del vettore di abilitazione e
     1     0     1     0

Si svuota il posto P1 dopo un tempo 40.000 (tempo tot=190)
Valore della marcatura continua corrente
0   180

vel1(debolmente abilitata)=0
vel2(non abilitata)=0

Valore della marcatura discreta corrente
1     0

Valore dei tempi residui di scatto Q (orologi)
   NaN
   NaN
    50
   NaN

Valore del vettore di abilitazione e
1     0     1     0

La transizione discreta che scatta al tempo 50
(tempo tot=240) è T3 (1 volta)

Valore della marcatura continua corrente
0   180

vel1(non abilitata)=0
vel2(fortemente abilitata)=2

Valore della marcatura discreta corrente
0     1

Valore dei tempi residui di scatto Q (orologi)
   NaN
```

```
   NaN
   NaN
    60


Valore del vettore di abilitazione e
     0     1     0     1
```

5. *Non ci resta che mandare in esecuzione il file di analisi,cioè*

$$analisi = analysis\_HPN(Evol, 1, 1, 1, 1, -1, 2, 1, 1, 1)$$

Otteniamo i seguenti risultati:

```
E[M(P1)]=47.500
E[M(P2)]=107.500
E[M(P3)]=0.750
E[M(P4)]=0.250
max(P1)=180.000
max(P2)=180.000
max(P3)=1.000
max(P4)=1.000
Prob[P3 = 0]=0.250
Prob[P3 = 1]=0.750
Prob[P4 = 0]=0.750
Prob[P4 = 1]=0.250
E[v1]=1.250
E[v2]=0.500
E[T3]=0.008
E[T4]=0.004
```

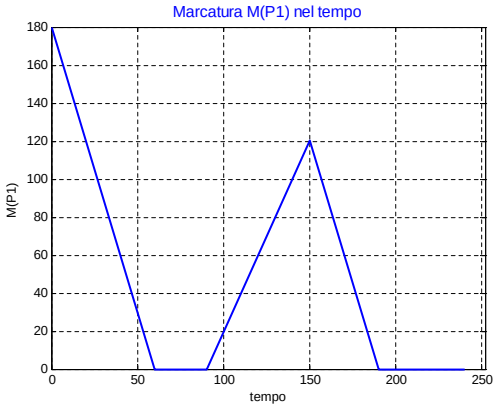


Figura 6.8: Marcatura nel posto p_1.

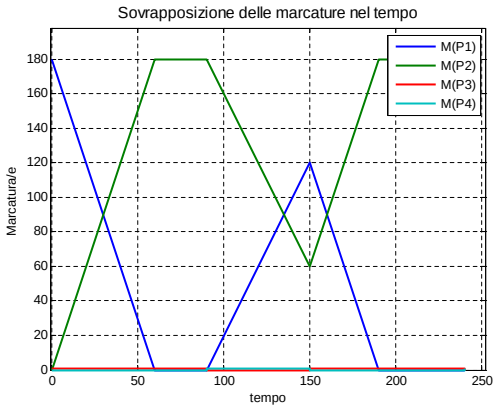


Figura 6.9: Sovrapposizione delle marcature nel tempo.

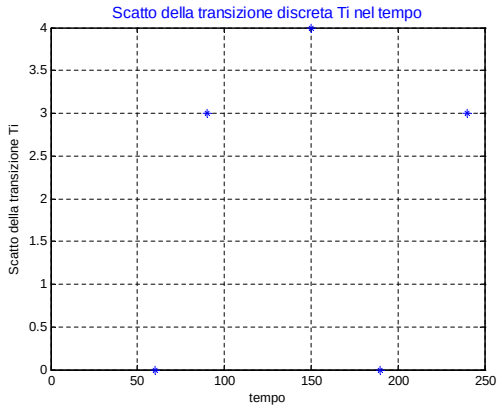


Figura 6.10: Scatto delle transizioni discrete.

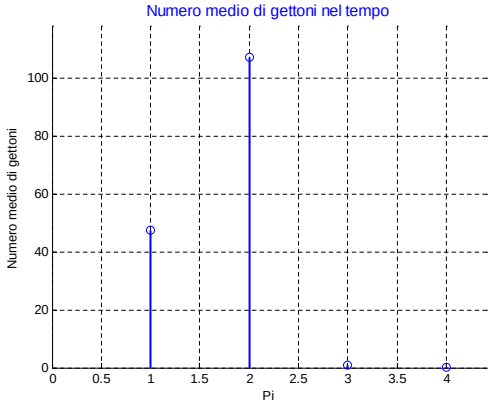


Figura 6.11: Numero medio di gettoni nei posti.

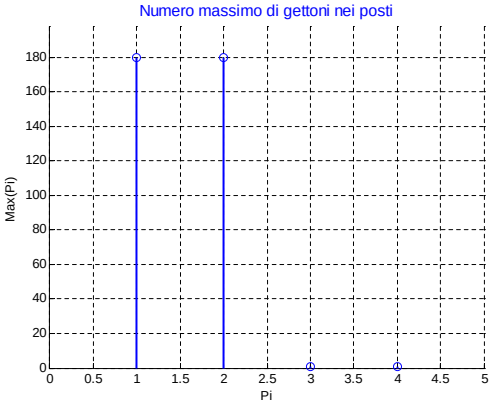


Figura 6.12: Numero massimo di gettoni nei posti.

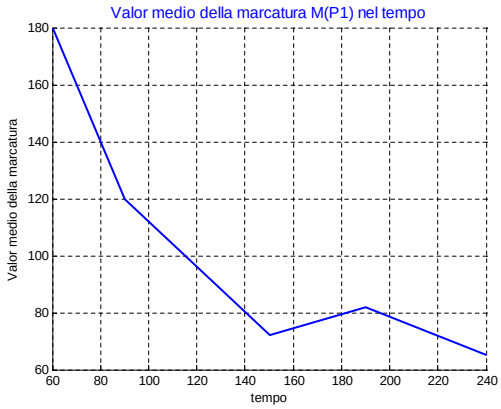


Figura 6.13: Valor medio della marcatura nel posto p_1.

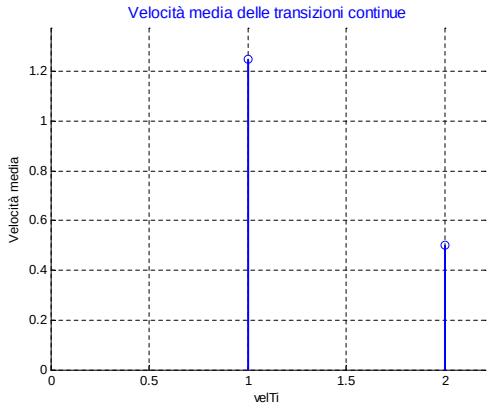


Figura 6.14: Velocità media delle transizioni continue.

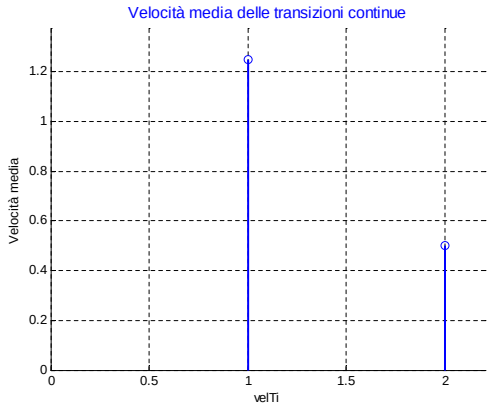


Figura 6.15: Frequenza di scatto delle transizioni discrete.

Appendice A

Distribuzioni di probabilità

A.1 Introduzione

Vengono qui introdotte le distribuzioni di probabilità più interessanti per le applicazioni alle reti di Petri: infatti è possibile attribuire alle transizioni discrete temporizzate dei tempi di scatto che non sono deterministici ma casuali dipendenti da una certa distribuzione. Il simulatore HYPENS è in grado di trattare le seguenti distribuzioni di probabilità:

1. Distribuzione esponenziale
2. Distribuzione uniforme
3. Distribuzione di Poisson
4. Distribuzione di Rayleigh
5. Distribuzione di Weibull
6. Distribuzione Beta
7. Distribuzione Chi Quadrato
8. Distribuzione del valore estremo
9. Distribuzione F
10. Distribuzione Gamma
11. Distribuzione del valore estremo generalizzato
12. Distribuzione di Pareto generalizzata

13. Distribuzione lognormale
14. Distribuzione F non centrale
15. Distribuzione t non centrale
16. Distribuzione Chi Quadrato non centrale
17. Distribuzione normale
18. Distribuzione t

Analizzeremo soltanto quelle che più utilizziamo nelle simulazioni, cioè le prime 5.

A.2 Le principali distribuzioni di probabilità

Le principali distribuzioni di probabilità che si utilizzano nelle reti di Petri sono: distribuzione esponenziale, distribuzione uniforme, distribuzione di Poisson, distribuzione di Rayleigh e distribuzione di Weibull. Vediamole nel dettaglio:

1. **La variabile casuale esponenziale** E' un caso particolare della variabile casuale gamma in cui il parametro p è posto uguale a 1. Il parametro λ della variabile casuale esponenziale corrisponde al secondo parametro della variabile casuale gamma. È spesso usata per modellare il tempo tra eventi indipendenti che avvengono con una frequenza media costante. In tabella A.1 sono riportati i principali parametri che caratterizzano la distribuzione e in Figura A.1 è rappresentato l'andamento della funzione di probabilità.

Parametri	$\lambda > 0$, detto *intensità*
supporto	$\mathbb{R}_+$
pdf	$f(x) = \lambda e^{-\lambda x}$
valore atteso	$\mu = \frac{1}{\lambda}$
varianza	$\sigma^2 = \lambda^{-2}$

Tabella A.1: Parametri della variabile casuale esponenziale.

2. **La variabile casuale uniforme** Si tratta di un caso particolare della variabile casuale Beta generalizzata, con p=q=1. In tabella A.2 sono riportati i principali parametri che caratterizzano la distribuzione e in Figura A.2 è rappresentato l'andamento della funzione di probabilità.

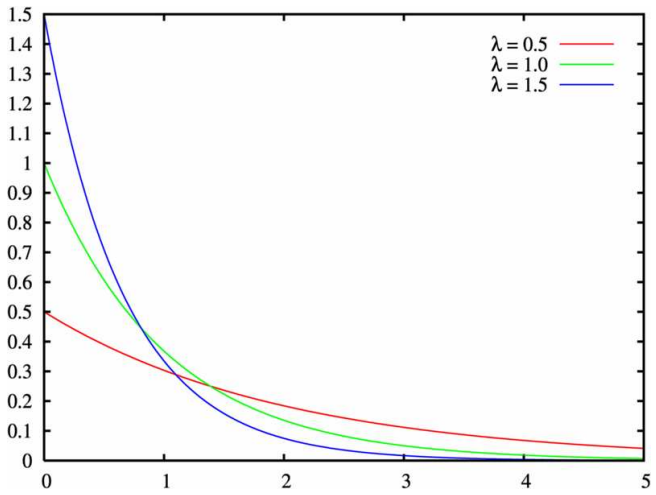


Figura A.1: Funzione di densità di probabilità della variabile casuale esponenziale.

Parametri	a, b
supporto	$\mathbb{R}_+$
pdf	$f(x) = \begin{cases} \frac{1}{(b-a)} & \text{se} \quad a \leq x \leq b \\ 0 & \text{se} \quad x < a, x > b \end{cases}$
valore atteso	$\mu = \frac{(a+b)}{2}$
varianza	$\sigma^2 = \frac{(b-a)^2}{12}$

Tabella A.2: Parametri della variabile casuale uniforme.

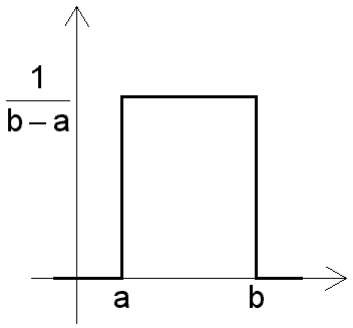


Figura A.2: Funzione di densità di probabilità della variabile casuale uniforme.

3. **La variabile casuale Poissoniana** La variabile casuale Poissoniana è definita con la funzione di probabilità riportata in tabella A.3 e rappresentata in Figura A.3. Si noti che λ è un qualsiasi valore positivo ($\lambda > 0$) equivalente al numero di successi che ci si aspetta e che si verificano in un dato intervallo di tempo. Per esempio, se un evento si verifica con una cadenza media di 4 minuti e vogliamo sapere quante volte questo evento si potrà verificare in 10 minuti, il valore di λ sarà $\lambda = \frac{10}{4} = 2,5$; e è la base del logaritmo naturale ($e = 2.71828\ldots$), x è il numero delle occorrenze (successi) per cui si vuole prevedere la probabilità.

Parametri	$\lambda \in \mathbb{R}_0$
supporto	$x \in \mathbb{N}$
funzione	$F(x) = \frac{e^{-\lambda}\lambda^x}{x!}$
valore atteso	$\mu = \lambda$
varianza	$\sigma^2 = \lambda$

Tabella A.3: Parametri della variabile casuale poissoniana.

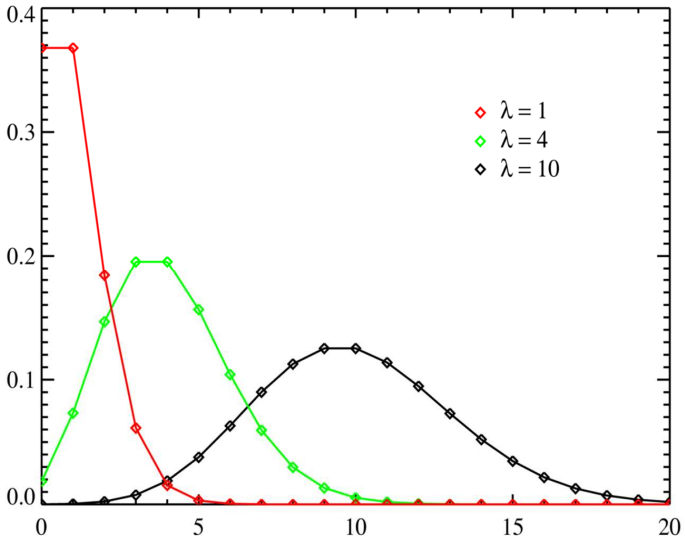


Figura A.3: Funzione di densità di probabilità di una variabile casuale poissoniana al variare di k (asse delle ordinate).

4. **La variabile casuale di Rayleigh** La variabile casuale di Rayleigh è una variabile casuale continua che si usa quando un vettore bidimensionale (come

ad esempio la velocità del vento) ha due componenti ortogonali e indipendenti distribuite secondo una variabile casuale normale. In tabella A.4 sono riportati i principali parametri che caratterizzano la distribuzione e in Figura A.4 è rappresentato l'andamento della funzione di probabilità.

Parametri	$\alpha \in \mathbb{R}^+$
supporto	$x \in \mathbb{R}_0^+$
pdf	$f(x,\alpha) = \frac{xexp\frac{-x^2}{2\alpha^2}}{\alpha^2}$
valore atteso	$\mu = \alpha\sqrt{\frac{(\pi)}{2}}$
varianza	$\sigma^2 = \frac{(4-\pi)}{2}\alpha^2$

Tabella A.4: Parametri della variabile casuale di Rayleigh.

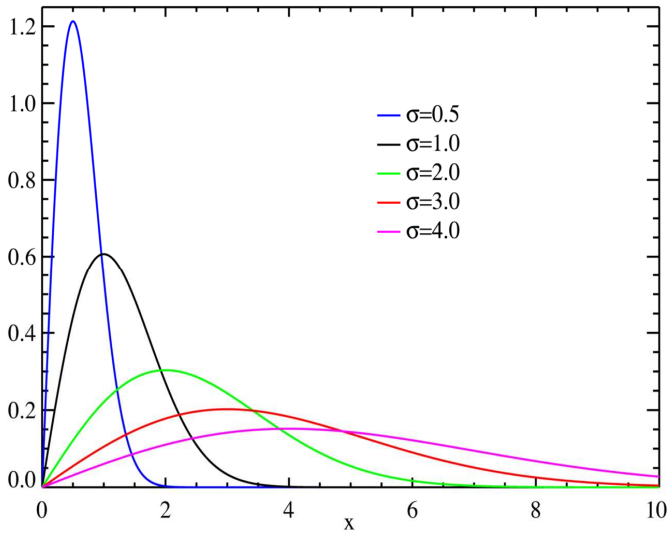


Figura A.4: Funzione di densità di probabilità di una variabile casuale di Rayleigh.

5. **La variabile casuale di Weibull** La variabile casuale di Weibull (dallo svedese Waloddi Weibull,1887-1979) è una variabile casuale continua utilizzata, per esempio, nell'ambito dei controlli di qualità industriali. Un esempio dell'applicazione della variabile casuale di Weibull è la descrizione della probabilità che una catena produttiva si fermi: il guasto di un suo membro porta all'arresto di tutta la catena. In tabella A.5 sono riportati i principali parametri che caratterizzano la distribuzione e in Figura A.5 è rappresentato l'andamento della funzione di probabilità. Si noti come per ilo calcolo

del valor medio e della varianza sia necessario l'utilizzo della funzione Γ, nota anche come funzione gamma di Eulero: essa è una funzione continua sui numeri reali positivi, che estende il concetto di fattoriale ai numeri complessi, nel senso che per ogni numero intero non negativo nd si ha

$$\Gamma(n+1) = n!$$

dove $n!$ è il fattoriale, cioè il prodotto dei numeri interi da 1 a nd:

$$n! = 1 * 2 * 3 \cdots * n$$

Parametri	$\lambda \in \mathbb{R}_0^+$ $k \in \mathbb{R}_0^+$
supporto	$x \in \mathbb{R}_0^+$
pdf	$f(x;k,\lambda) = \frac{k}{\lambda}\frac{x}{\lambda}^{(k-1)} e - (\frac{x}{\lambda})^k$
valore atteso	$\mu = \lambda\Gamma(1+\frac{1}{k}$
varianza	$\sigma^2 = \lambda(\frac{k-1}{k})^{\frac{1}{k}}$ se $k > 1$

Tabella A.5: Parametri della variabile casuale di Weibull.

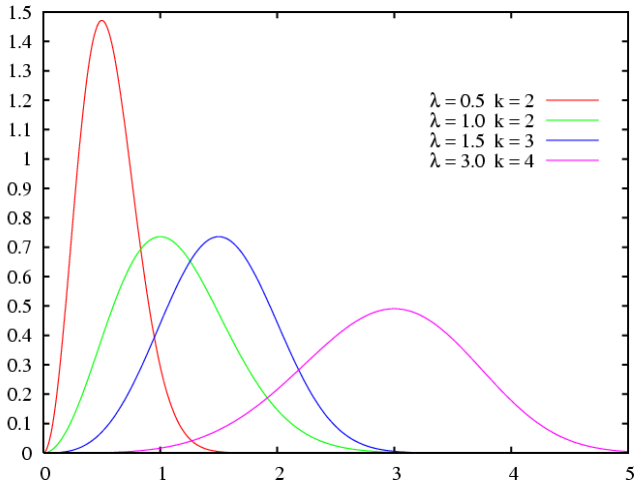


Figura A.5: Funzione di densità di probabilità di una variabile casuale di Weibull.

Bibliografia

[1] F. Balduzzi, A. Giua, and C. Seatzu. Modeling and simulation of manufacturing systems with first-order hybrid petri nets.

[2] R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2005.

[3] A. Di Febbraro and A. Giua. *Sistemi a eventi discreti*. McGraw-Hill, 2002.