Università degli Studi di Cagliari
Facoltà di Ingegneria
Dipartimento di Ingegneria Elettrica ed Elettronica
Corso di Laurea Specialistica in Ingegneria Elettronica

Applicazione di tecniche di Infinitesimal Perturbation Analysis alle reti di Petri ibride

Relatori:
Prof. Alessandro Giua
Prof.ssa Carla Seatzu

Tesi di Laurea di:
Davide Nessi

Anno Accademico 2009/2010

Qual è 'l geomètra che tutto s'affige
per misurar lo cerchio, e non ritrova,
pensando, quel principio ond'elli indige,

tal era io a quella vista nova:
veder voleva come si convenne
l'imago al cerchio e come vi s'indova;

ma non eran da ciò le proprie penne:
se non che la mia mente fu percossa
da un fulgore in che sua voglia venne.

A l'alta fantasia qui mancò possa;
ma già volgeva il mio disio e 'l velle,
sì come rota ch'igualmente è mossa,

l'amor che move il sole e l'altre stelle.

Dante Alighieri, Divina Commedia, Paradiso, Canto 33
(vv. 133-145).

Alla mia famiglia.

Indice

1 Introduzione **2**
- 1.1 Motivazione e letteratura 2
- 1.2 Oggetto della tesi 5
- 1.3 Contributi originali 6
- 1.4 Risultati ottenuti 7
- 1.5 Descrizione della tesi 7

2 Richiami sulle Reti di Petri **8**
- 2.1 Reti di Petri posto/transizione 9
  - 2.1.1 Marcatura 10
  - 2.1.2 Abilitazione e scatto 10
  - 2.1.3 Reti di Petri temporizzate 12
  - 2.1.4 Conflitto 15
- 2.2 Reti di Petri Ibride 18
  - 2.2.1 Struttura e marcatura 19
  - 2.2.2 Abilitazione e scatto 21
  - 2.2.3 Dinamica della rete 22
  - 2.2.4 Vettori di velocità istantanea di scatto ammissibili . . 25
  - 2.2.5 Conflitto 27
  - 2.2.6 Risoluzione conflitto globale 29
  - 2.2.7 Risoluzione conflitto locale 30

3 Il simulatore HYPENS **32**
- 3.1 Il linguaggio di programmazione MATLAB 32
- 3.2 Caratteristiche di HYPENS 33
- 3.3 Funzione enter_HPN 34
  - 3.3.1 Dati in ingresso 34
  - 3.3.2 Dati restituiti in uscita 37
- 3.4 Funzione make_HPN.m 39
  - 3.4.1 Dati in ingresso 39
  - 3.4.2 Dati restituiti in uscita. 41
- 3.5 Funzione simulator_HPN.m 41
  - 3.5.1 Dati in ingresso 41

3.5.2 Dati restituiti in uscita 42
3.6 Funzione analysis.m 43
3.6.1 Dati in ingresso 44
3.6.2 Dati in uscita 46
3.7 Estensione del simulatore ed esempio applicativo 47
3.7.1 simulator1_HPN 47
3.7.2 Simulazione impianto di imbottigliamento 48

4 IPA per reti di Petri ibride **51**
4.1 Infinitesimal Perturbation Analysis 51
4.2 Sensitività del flusso utile nelle transizioni continue 52
4.3 Modello rete manifatturiera con tre macchine 56
4.4 Rete manifatturiera con n macchine 59
4.5 IPA per l'ottimizzazione delle dimensioni di un buffer 60
4.6 Stima del gradiente delle funzioni obbiettivo 65
4.7 Realizzazione modello tramite rete di Petri ibrida 68

5 Esempi applicativi di IPA per reti di Petri ibride **70**
5.1 Simulazione rete manifatturiera con tre macchine 70
5.2 Simulazione buffer 71
5.2.1 Simulazione buffer: *inflow rate* e *outflow rate* costanti 74
5.2.2 Simulazione buffer: *inflow rate* costante, *service rate* periodico 76
5.2.3 Simulazione buffer: *inflow rate burst*, *outflow rate* costante 77
5.2.4 Stima gradiente della perdita e della lunghezza media della coda 79
5.3 Ottimizzazione parametri di *routing* 82
5.4 Pesi della funzione obbiettivo variabili 82

6 Conclusioni **85**

A Codice **87**
A.1 simulator1_HPN 87
A.2 Funzione per il controllo del buffer con beta costante 88
A.3 Funzione per il controllo del buffer con beta variabile 89
A.4 Funzione per il controllo del buffer con traffico *burst* 90
A.5 Funzione per il controllo del buffer con traffico casuale 91
A.6 IPA_buffer(x,theta,dt) 92
A.7 Funzione per la generazione dei vincoli nella simulazione della rete manifatturiera 93
A.8 Funzione per la simulazione della rete di imbottigliamento . . 94
A.9 Funzione per l'ottimizzazione dei parametri di routing 96

Bibliografia **99**

Ringraziamenti

Voglio ringraziare tutte le persone che mi sono state particolarmente vicine in quest'ultima fase e forse più difficile della mia esperienza universitaria. In particolare la mia famiglia che mi ha sempre spronato a trovare dentro di me le energie per realizzare questa tesi. Senza di loro non sarei riuscito a realizzare questo lavoro. Alessandro Giua, Carla Seatzu, Maria Paola Cabasino sono stati sempre molto disponibili e benevoli nei miei confronti e hanno sempre chiarito qualsiasi mio dubbio. Maria Strocchia mi è stata particolarmente vicina nei momenti più duri infondendomi coraggio e supporto morale. Ringrazio Paolo Pintus per l'amicizia dimostratami, il supporto tecnico e morale. Nicola Murgia mi è stato amico e collega di studio per tutto il mio percorso universitario. Domenico Saiu, Matteo Sollai, Damiano Locci sono tra i miei amici scacchisti quelli con cui ho giocato più partite, con i quali ho trascorso ore spensierate, ma con cui ho anche avuto occasione di conoscere a fondo il mio carattere e crescere umanamente.

Capitolo 1

Introduzione

In questo capitolo intendiamo discutere in maniera informale i modelli e le tecniche utilizzate in questa tesi. Si introdurranno le reti di Petri, continue, ibride e se ne evidenzieranno i vantaggi come modello; verrà poi introdotta la tecnica dell' analisi perturbativa infinitesimale (*Infinitesimal Perturbation Analysis*) e la sua possibilità di applicazione alle reti di Petri ibride. Si farà una rassegna della letteratura prodotta e infine si descriveranno brevemente i contributi originali della tesi.

1.1 Motivazione e letteratura

Le reti di Petri sono un modello ad eventi discreti proposto inizialmente da C. A. Petri nella sua tesi di dottorato [12]. La principale caratteristica di una rete di Petri è che il suo stato è un vettore di interi non negativi. Questo è il principale vantaggio rispetto ad altri formalismi come gli automi, dove lo spazio di stato è un insieme non-strutturato simbolico, ed è stato utilizzato per sviluppare molte tecniche di analisi che non richiedono di enumerare lo spazio di stato (analisi strutturale) [1]. Un'altra caratteristica chiave delle reti di Petri è la loro capacità di rappresentare graficamente e visualizzare primitive come parallelismo, concorrenza, sincronizzazione, mutua esclusione, etc.

Nella relativa letteratura sono state proposte varie estensioni delle reti di Petri. In questa tesi focalizzeremo l'attenzione sulle reti di Petri ibride. Descriviamo brevemente alcune delle estensioni delle reti di Petri che si trovano in letteratura.

- Reti di Petri continue: nascono dall'esigenza di ridurre la complessità computazionale dell'analisi e ottimizzazione di problemi su scala reale, i quali diventano velocemente intrattabili sia analiticamente che computazionalmente con i modelli ad eventi discreti. Ad esempio, questo accade spesso in alcuni sistemi produttivi dove il numero di parti in

un *buffer* può essere molto grande, tanto da essere conveniente approssimarlo con un numero reale. In effetti, l'idea di fluidificazione (*fluidification*) è stata prima applicata ad altri modelli ad eventi discreti come gli automi. In seguito, è stata applicata con successo alle reti di Petri ibride [6], [14].

I principali vantaggi nell'usare l'approssimazione fluida per l'analisi ed il controllo possono essere riassunti come segue.

1. Vi è la possibilità di un considerevole incremento nella efficienza computazionale, perché la simulazione di modelli fluidi può essere spesso eseguita molto più efficientemente rispetto all sua parte discreta.
2. L'approssimazione fluida consente di ridurre la dimensione dello spazio di stato.
3. I parametri di dimensionamento nei modelli fluidi sono continui, quindi è possibile usare l'informazione del gradiente per rendere più veloce l'ottimizzazione e per migliorare l'analisi di sensitività.

In generale, la sola approssimazione fluida non è sufficiente a descrivere un sistema in maniera conveniente. Ad esempio nel dominio manifatturiero le macchine possono essere operative o no, i *buffer* pieni o vuoti e così via. Quindi, il sistema risultante può essere meglio descritto da un modello ibrido, dove la parte discreta descrive l'operatività o meno delle macchine e differenti dinamiche continue sono associate a ciascun stato discreto.

- Reti di Petri ibride: mantengono tutte quelle caratteristiche che fanno delle reti di Petri un ottimo modello ad eventi discreti: non richiedono una enumerazione esaustiva dello spazio di stato e possono descrivere sistemi con un infinito spazio di stato; permettono una rappresentazione modulare, dove la struttura di ciascun modulo è mantenuta nel modello composto; lo spazio discreto è rappresentato da un vettore e non da una etichetta simbolica, quindi possono essere utilizzate tecniche di algebra lineare per la loro analisi.

Le reti di Petri ibride sono dunque una classe di modelli che combinano dinamiche discrete e continue. Esse estendono il tradizionale concetto di rete di Petri per includere il flusso di fluido.

Il formalismo è stato introdotto in [6], [14] e, con alcune revisioni riguardo l'abilitazione delle transizioni continue, in [4]. Per descrivere il comportamento di una rete di Petri ibrida si introduce il concetto di macro-stato e macro-evento. Ciascun macro-stato è caratterizzato:

1. dalla marcatura discreta della rete;

2. dall'insieme dei posti continui vuoti.

Tipicamente si distinguono due tipi di macro-eventi:

1. una transizione discreta scatta;

2. un posto continuo si svuota.

Sessego in [15] distingue quattro tipi di macro-eventi considerando, oltre ai due citati sopra, l'abilitazione o disabilitazione di una transizione discreta temporizzata in funzione della marcatura dei posti continui.

Come in tutti i modelli ibridi si distinguono due livelli di comportamento:

1. a livello più basso, l'evoluzione continua della rete è descritta da modelli fluidi del primo ordine;

2. a livello più alto, un modello ad eventi discreti descrive il macro-comportamento della rete che, al verificarsi dei macro-eventi, evolve attraverso una sequenza di macro-stati.

Il campo di applicazione del modello riguarda soprattutto il dominio manifatturiero come ad esempio impianti di imbottigliamento [13]. Le transizioni continue sono caratterizzate da una velocità minima (*minimum firing speed*, mfs), una massima (*maximum firing speed*, MFS) e la loro velocità istantanea deve appartenere ad un insieme ammissibile ossia rispettare dei vincoli. I vincoli possono essere così riassunti:

- la velocità istantanea di ciascuna transizione deve essere compresa tra quella minima e quella massima;

- poiché nessun posto può avere una marcatura negativa tutti i posti continui che sono vuoti all'inizio di un macro-evento dovranno avere un bilanciamento non negativo;

- in [13] vengono introdotti, per esigenze di modellizzazione, dei posti continui la cui marcatura è vincolata a rimanere nulla.

In generale il calcolo di un vettore di velocità istantanee (*istantaneous firing speed*) non è un compito semplice. In [6], viene proposto un algoritmo iterativo per determinare un singolo vettore delle velocità istantanee; l'algoritmo massimizza le velocità delle transizioni continue rispettando delle regole di priorità. In [4], viene seguito invece un approccio differente. Viene caratterizzato per mezzo di disuguaglianze lineari l'insieme $\mathcal{S}$ di tutti i vettori delle velocità istantanee ammissibili. Ciascun vettore $\mathbf{v} \in \mathcal{S}$ rappresenta un modo operativo particolare del sistema descritto dalla rete e, tra tutti i possibili modi di operare, è possibile scegliere il migliore in accordo con un dato obbiettivo (si tratterà di massimizzare una funzione $\mathbf{c}^T \cdot \mathbf{v}$). Il simulatore *HYPENS* (sviluppato da Sessego [15]) si basa su questo tipo di

approccio. Questo tipo di ottimizzazione è miopica nel senso che si ottiene una soluzione ottima solo in un macro-stato.

Per superare il problema dell'ottimizzazione miopica recentemente si è studiata la possibilità di applicare tecniche di analisi perturbativa infinitesimale (*Infinitesimal Perturbation Analysis*) alle reti di Petri ibride [2]. L'analisi perturbativa infinitesimale è stata applicata con successo ai modelli di fluido stocastici (*Stochastic Flow Model*) in [17], [5], [3]. L'analisi perturbativa infinitesimale è una tecnica di ottimizzazione che si basa su una stima del gradiente della funzione obbiettivo. In [2] e [5] sono proposti algoritmi e formule per la stima del gradiente della funzione obbiettivo.

1.2 Oggetto della tesi

Questa tesi ha come oggetto di studio l'analisi perturbativa infinitesimale, la sua possibilità di applicazione alle reti di Petri ibride per l'ottimizzazione e il confronto con una tecnica di natura euristica da me ideata e sviluppata in questa tesi. Sono presenti in letteratura alcune formule e algoritmi, di facile implementazione, che consentono di stimare, simulando l'evoluzione di un modello fluido stocastico, i gradienti delle funzioni di prestazione. Essendo le reti di Petri ibride un ottimo modello fluido stocastico e il *tool* HYPENS un simulatore per queste reti è auspicabile poterlo utilizzare per i fini sopra descritti, evitando lo sviluppo di simulatori *ad-hoc* e cercando di perseguire il riuso del *software*. HYPENS presentava alcuni limiti che impedivano potesse essere impiegato immediatamente per la stima dei gradienti delle funzioni di prestazione e per l'ottimizzazione. I limiti possono essere così sintetizzati:

- non è possibile vincolare l'evoluzione della rete in funzione dei parametri da cui dipende la funzione obbiettivo;
- non è possibile specificare ulteriori vincoli per le velocità delle transizioni continui oltre a quelli presenti nell'insieme $\mathcal{S}$.

Una prima modifica apportata riguarda quindi l'evoluzione della parte continua e consente di utilizzare il *tool* in maniera più generale dando l'opportunità all'utente di specificare ulteriori vincoli oltre a quelli presenti nell'insieme $\mathcal{S}$, ad esempio specificando in forma chiusa la velocità delle transizioni continue in funzione della marcatura raggiunta dalla rete o vincolare la marcatura di un posto continuo a rimanere costante durante tutta l'evoluzione. Sarà possibile ad esempio vincolare l'evoluzione della rete a dei parametri (che in letteratura vengono solitamente indicati con θ) e dai quali dipende una funzione di prestazione (solitamente indicata $J(\theta)$). Tali parametri possono rappresentare ad esempio vincoli di *routing* [2]. Le velocità delle transizioni continue potranno essere vincolate a seguire un andamento deterministico periodico oppure uno che soddisfi alcuni vincoli statistici. In questo modo sarà possibile ad esempio modellare e simulare svariati tipi di sorgenti.

Le potenzialità del *tool* sono dunque notevolmente ampliate consentendo di simulare l'evoluzione di una vasta gamma di sistemi che risultano più facilmente modellabile grazie alle reti di Petri ibride e i cui parametri di interesse possono essere ottimizzati tramite una stima dei gradienti delle funzioni di prestazione. Si pensi ad esempio ad una rete di telecomunicazione in cui i pacchetti viaggiano e vengono memorizzati in un *buffer*. Ovviamente sarebbe possibile modellare tale sistema con una rete di Petri ma in una rete di telecomunicazione reale il numero di pacchetti è talmente elevato che risulta molto più semplice utilizzare un modello continuo. In letteratura esistono numerosi esempi [2], [5] di come sia possibile stimare il gradiente di una funzione di prestazione nota l'evoluzione della marcatura della rete per un intervallo temporale fissato. Le stime avvengono tramite algoritmi o formule facilmente implementabili.

Una seconda modifica apportata al software nasce dall'esigenza di sperimentare la tecnica di natura euristica menzionata precedentemente. Sarà possibile, in funzione della marcatura raggiunta dalla rete, dare maggiore priorità ad alcune transizioni. Si sperimenterà quindi un nuovo tipo di approccio per la massimizzazione globale. I pesi della funzione obbiettivo (il vettore $\mathbf{c}$) verranno scelti in maniera euristica, ad ogni macro evento, in funzione della marcatura raggiunta dalla rete. E' facile convincersi già da ora che una tecnica simile avrà degli svantaggi rispetto all'ottimizzazione tramite analisi perturbativa infinitesimale.

1. Non è una tecnica sistematica a differenza dell'ottimizzazione basata sull'analisi perturbativa infinitesimale.

2. Non si è trovata in letteratura alcun riferimento ad una tecnica simile.

Malgrado ci siano questi svantaggi la tecnica potrebbe dimostrarsi utile in reti dove non siano disponibili algoritmi e formule per la stima del gradiente della funzione obbiettivo.

1.3 Contributi originali

I contributi originali di questa tesi possono essere così riassunti:

- si è studiata la possibilità di applicazione dell'analisi perturbativa infinitesimale alle reti di Petri ibride per l'ottimizzazione;

- si sono studiate alcune reti di particolare interesse applicativo, le formule necessarie per la stima del gradiente delle funzioni di prestazione e sono state generalizzate alcune di queste formule;

- si sono studiati gli algoritmi che forniscono la stima delle funzioni di prestazione e si è fornita una implementazione in codice MATLAB;

- si è modificato opportunamente il *tool* HYPENS affinché gli algoritmi per la stima del gradiente della funzione di prestazione potessero essere utilizzati per l'ottimizzazione;
- si sono effettuate delle simulazioni per verificarne il corretto funzionamento.

1.4 Risultati ottenuti

Dalle simulazioni effettuate si è verificata la correttezza dell'evoluzione di alcune reti. Grazie agli algoritmi implementati per la stima del gradiente delle funzioni di prestazione, è stato possibile utilizzare *HYPENS* per l'ottimizzazione dei parametri. Il tentativo di risolvere il problema dell'ottimizzazione miopica facendo variare i pesi della funzione obbiettivo (vettore $\mathbf{c}$), ad ogni macro-periodo, in funzione della marcatura raggiunta dalla rete non ha dato buoni risultati; si riescono a raggiungere prestazioni paragonabili a quelle ottenibili con l'IPA ma lo svantaggio principale consiste che non si sono trovate tecniche sistematiche per la scelta del vettore $\mathbf{c}$.

1.5 Descrizione della tesi

Nel capitolo 2 si faranno dei brevi richiami ai modelli matematici delle reti di Petri discrete, continue e ibride. In particolare, per le reti di Petri ibride, verrà richiamato il concetto di *Istantaneous Firing Speed* (IFS), di macro-evento e di macro-periodo [4]. Verranno proposti alcuni esempi per chiarire le definizioni formali.

Nel capitolo 3 si descriverà il *tool* HYPENS utilizzato per le simulazioni, verranno descritte le varie funzioni da cui è formato, i parametri in ingresso e in uscita a ciascuna funzione.

Nel capitolo 4 verrà introdotta la tecnica dell'analisi perturbativa infinitesimale e si vedrà come essa trovi naturale applicazione nelle reti di Petri Ibride. Si descriveranno gli algoritmi e le formule per la stima del gradiente delle funzioni di prestazione.

Nel capitolo 5 si descriveranno le modifiche apportate al *tool* e verranno presentati alcuni esempi di applicazione del *tool* modificato ad alcune reti di particolare interesse:

1. una rete manifatturiera;
2. un buffer;
3. un impianto di imbottigliamento.

Capitolo 2

Richiami sulle Reti di Petri

Fra i vari modelli ad eventi discreti, le Reti di Petri hanno una importanza predominante a causa di vari fattori:

- Sono sia un formalismo grafico che un formalismo matematico. Come 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 per studiare le proprietà di interesse.
- Permettono di dare una rappresentazione compatta di sistemi con un grande spazio di stato.
- Permettono di rappresentare esplicitamente il concetto di concorrenza, cioè di attività che possono essere eseguite parallelamente.
- Consentono una rappresentazione modulare, cioè se un sistema è composto da più sottosistemi che interagiscono tra loro, è generalmente possibile rappresentare ciascun sottosistema con una sotto-rete che verrà poi integrata con opportuni operatori di rete per ottenere il sistema complessivo.

Le reti di Petri si suddividono inoltre in *logiche* e *temporizzate*; le prime non consentono di rappresentare l'evoluzione temporale degli eventi, ma solo l'ordine con cui essi si verificano. Le reti di Petri temporizzate si suddividono inoltre in deterministiche e stocastiche. Le reti di Petri continue sono la versione fluidizzata delle reti di Petri.

- La marcatura dei posti è un numero reale.
- Il peso degli archi è un numero reale.
- Le transizioni sono caratterizzate da una velocità di scatto.

Le reti di Petri ibride sono costituite sia da una parte discreta che da una parte continua.

2.1 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* (rappresentate da barre o da rettangoli). Gli archi, che connettono i posti alle transizioni e viceversa, devono essere orientati.

Definizione 2.1 *: Una rete di Petri P/T è una struttura* $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ *dove:*

- $P = \{p_1, p_2, \cdots, p_m\}$ *è l'insieme di* m *posti.*
- $T = \{t_1, t_2, \cdots, t_n\}$ *è l'insieme di* n *transizioni.*
- $\boldsymbol{Pre} : P \times T \to \mathbb{N}$ *è la funzione di* pre-incidenza *che specifica gli archi diretti dai posti alle transizioni e viene rappresentata tramite una matrice* $m \times n$*. In particolare* $Pre(p_i, t_j)$ *è il numero di archi che connettono il posto* p_i *alla transizione* t_j*.*
- $\boldsymbol{Post} : P \times T \to \mathbb{N}$ *è la funzione di* post-incidenza *che specifica gli archi diretti dalle transizioni ai posti e viene rappresentata tramite una matrice* $m \times n$*. In particolare* $Post(p_i, t_j)$ *è il numero di archi che connettono la transizione* t_j *al posto* p_i*.*

Si noti che nelle reti di Petri P/T le matrici **Pre** e **Post** sono delle matrici di interi non negativi. Si indica con $\mathbf{Pre}(p_i, \cdot)$ la riga relativa al posto p_i e con $\mathbf{Pre}(\cdot, t_i)$ la colonna relativa alla transizione t_i. La stessa notazione vale per la matrice **Post**. Si ipotizza che $P \cap T = \emptyset$ e che $P \cup T \neq \emptyset$ cioè la rete è costituita da almeno un posto o da una transizione.

E' utile introdurre un' altra matrice, detta matrice di incidenza, che è così definita:

$$\mathbf{C} = \mathbf{Post} - \mathbf{Pre}. \tag{2.1}$$

Data la matrice di incidenza non è sempre possibile ricostruire il grafo mentre, date le matrici di **Pre** e **Post**, questo è sempre possibile.

Data una transizione t_j è possibile definire i seguenti insiemi di posti:

- Insieme dei posti in ingresso alla transizione t_j
  ${}^\bullet t_j = \{p_i \in P | Pre(p_i, t_j) > 0\}$.
- Insieme dei posti in uscita alla transizione t_j
  $t_j^\bullet = \{p_i \in P | Post(p_i, t_j) > 0\}$.

Dato un posto p_i è possibile definire i seguenti insiemi di transizioni :

- Insieme delle transizioni in ingresso al posto p_i
  ${}^\bullet p_i = \{t_j \in T | Post(p_i, t_j) > 0\}$.
- Insieme dei posti in uscita al posto p_i
  $p_i^\bullet = \{t_j \in T | Pre(p_i, t_j) > 0\}$.

2.1.1 Marcatura

E' una funzione tramite la quale è possibile definire lo stato di una rete P/T.

Definizione 2.2 *Una* marcatura *è una funzione* $\boldsymbol{M} : P \to \mathbb{N}$ *che assegna ad ogni posto un numero intero non negativo di marche (gettoni) rappresentate graficamente con dei pallini neri dentro i posti. Una rete* N *con una marcatura iniziale* $\boldsymbol{M}_0$ *viene detta rete marcata e viene indicata con* $\langle N, \boldsymbol{M}_0 \rangle$.

2.1.2 Abilitazione e scatto

Definizione 2.3 *Una transizione* t_j *è abilitata dalla marcatura* $\mathbf{M}$ *se*

$$\boldsymbol{M} \geq \boldsymbol{Pre}(\cdot, t_j) \tag{2.2}$$

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

Per indicare che t è abilitata da $\mathbf{M}$ si scrive $\mathbf{M}[t\rangle$. Dall'ispezione visiva della rete è facile rendersi conto se una transizione t è abilitata: occorre che ogni posto in ingresso alla transizione, cioè ogni posto $p \in^\bullet t$ abbia un numero di marche pari a superiore agli archi *pre* che vanno dal posto alla transizione.

Una transizione t_j abilitata dalla marcatura $\mathbf{M}$ 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 $\mathbf{M'}$:

$$\mathbf{M'} = \mathbf{M} - \mathbf{Pre}(\cdot, t_j) + \mathbf{Post}(\cdot, t_j) = \mathbf{M} + \mathbf{C}(\cdot, t_j). \tag{2.3}$$

Per indicare che lo scatto della transizione t_j porta la rete dalla marcatura $\mathbf{M}$ alla marcatura $\mathbf{M'}$ si scrive: $\mathbf{M}[t_j\rangle\mathbf{M'}$.

Una sequenza di transizioni $t_x, t_y, \cdots, t_z,\ t_x \in T, t_y \in T \cdots t_z \in T$ è abilitata da una marcatura $\mathbf{M}$ se t_x è abilitata da $\mathbf{M}$ e il suo scatto porta a $\mathbf{M}_1 = \mathbf{M} + \mathbf{C}(\cdot, t_x)$; se la transizione t_y è abilitata da $\mathbf{M}_1$ e il suo scatto porta a $\mathbf{M}_2 = \mathbf{M}_1 + \mathbf{C}(\cdot, t_y)$ etc. Una sequenza $\boldsymbol{\sigma}$ viene anche detta sequenza di scatto alla quale corrisponde la traiettoria: $\mathbf{M}[t_x\rangle\mathbf{M}_1[t_y\rangle\mathbf{M}_2 \cdots [t_z\rangle\mathbf{M}_r$. Per indicare che la sequenza $\boldsymbol{\sigma}$ è abilitata da $\mathbf{M}$ si scrive $\mathbf{M}[\boldsymbol{\sigma}\rangle$. Per indicare che lo scatto di $\boldsymbol{\sigma}$, a partire dalla marcatura $\mathbf{M}$, determina la marcatura $\mathbf{M'}$ si scrive: $\mathbf{M}[\boldsymbol{\sigma}\rangle\mathbf{M'}$.

Data una rete marcata $\langle N, \mathbf{M}_0 \rangle$ e una sequenza di transizioni $\boldsymbol{\sigma}$ si definisce vettore caratteristico $\boldsymbol{\sigma} = [\sigma_1\sigma_2 \cdots \sigma_n]^T$, il vettore caratteristico di $\boldsymbol{\sigma}$ la cui generica componente $\boldsymbol{\sigma}_j = |\boldsymbol{\sigma}|_{t_j}$ indica quante volte la transizione t_j compare in $\boldsymbol{\sigma}$.

In una rete marcata N, con matrice di incidenza $\mathbf{C}$, se una marcatura $\mathbf{M}$ è raggiungibile da $\mathbf{M}_0$ con $\boldsymbol{\sigma}$ allora possiamo scrivere:

$$\mathbf{M} = \mathbf{M}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}. \tag{2.4}$$

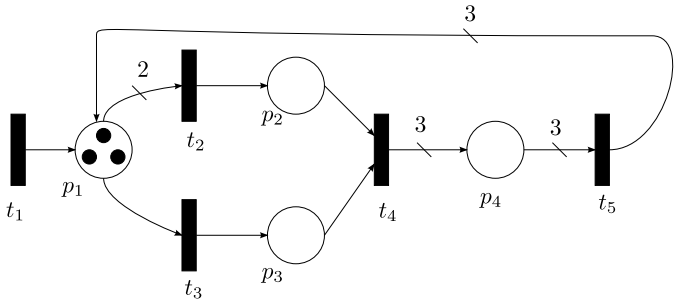


Figura 2.1: Rete di Petri.

Esempio 2.4 *In figura 2.1 è mostrato un esempio di rete di Petri P/T marcata. L'insieme dei posti e delle transizioni sono:*

$$P = \{p_1, p_2, p_3, p_4\}$$

$$T = \{t_1, t_2, t_3, t_4, t_5\}.$$

Le matrici ***Pre*** *e* ***Post*** *sono le seguenti:*

$$\boldsymbol{Pre} = \begin{bmatrix} 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

$$\boldsymbol{Post} = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \end{bmatrix}.$$

Da cui si ricava la matrice di incidenza:

$$\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre} = \begin{bmatrix} 1 & -2 & -1 & 0 & 3 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 3 & -3 \end{bmatrix}.$$

Si noti che in questo caso, è possibile ricostruire il grafo direttamente dalla matrice $\boldsymbol{C}$*. La marcatura iniziale è data dal vettore:*

$$\boldsymbol{M}_0 = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Si noti che la transizione t_1 è sempre abilitata poiché vale sempre la 2.2. Una possibile sequenza di transizioni abilitate dalla marcatura $\boldsymbol{M}_0$ è la seguente:

$$t_2, t_3, t_4, t_5$$

a cui corrisponde la seguente traiettoria (facilmente ottenibile per ispezione visiva):

$$\begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \end{bmatrix} [t_2\rangle \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} [t_3\rangle \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} [t_4\rangle \begin{bmatrix} 0 \\ 0 \\ 0 \\ 3 \end{bmatrix} [t_5\rangle \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Si poteva ottenere lo stesso risultato definendo il vettore caratteristico

$$\boldsymbol{\sigma} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \end{bmatrix}^T$$

ottenendo:

$$\boldsymbol{M} = \boldsymbol{M}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -2 & -1 & 0 & 3 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 3 & -3 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Nel caso volessimo conoscere il valore della marcatura dopo lo scatto della sequenza t_2, t_3 avremmo definito come vettore di scatto

$$\boldsymbol{\sigma} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

ottenendo correttamente:

$$\boldsymbol{M} = \boldsymbol{M}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -2 & -1 & 0 & 3 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 3 & -3 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

2.1.3 Reti di Petri temporizzate

Al fine di rendere le reti di Petri uno strumento utile per l'analisi quantitativa delle prestazioni è necessario introdurre una qualche metrica temporale. Esse offrono diversi vantaggi che possono essere sintetizzati nei punti seguenti:

- Consentono di modellare la stocasticità intrinseca dei sistemi reali.
- Possono essere utilizzate sia come modelli logici che prestazionali.

- Possono riprodurre le caratteristiche proprie dei sistemi ad eventi discreti quali priorità, sincronizzazione.
- Possono essere analizzate per mezzo di strumenti informatici dedicati.

Una rete di Petri $\langle N, \mathbf{M}_0 \rangle$ temporizzata è data dalla coppia $\langle R, Tempo \rangle$ dove:

- R è una rete di Petri marcata
- $Tempo : T \to \mathbb{R}_0^+$ è una funzione associata all'insieme di transizioni T secondo la quale $Tempo(t_j) = d_j$ corrisponde al tempo associato a t_j.

Lo scatto viene considerato un' operazione indivisibile e la temporizzazione è realizzata associando ad ogni transizione un ritardo che corrisponde al tempo che deve trascorrere fra la sua abilitazione e il successivo scatto. I gettoni rimangono nei posti in ingresso a ciascuna transizione sino allo scatto a meno che un altra transizione non li prelevi a sua volta. Solo se, al termine del ritardo d_j, la condizione di abilitazione continua a sussistere, allora la transizione t_j scatta effettivamente facendo si che i gettoni siano prelevati dai posti in ingresso e depositati nei posti in uscita.

Una transizione t_j è detta:

- *immediata* se scatta appena viene abilitata;
- *deterministica* se ad essa è associato un ritardo di scatto d_j scelto in maniera deterministica;
- *stocastica* se il tempo di ritardo d_j è una variabile aleatoria con funzione di probabilità nota.

Data una marcatura $\mathbf{M}$ si definsce grado di abilitazione della transizione t_j

$$enab(\mathbf{M}, t_j) = max\{k \in \mathbb{N} | \mathbf{M} \geq k\mathbf{Pre}(\cdot, t_j)\}. \tag{2.5}$$

Questo significa che la transizione t_j avente un ritardo d_j a partire dall'istante τ_j potrebbe scattare $enab(\mathbf{M}, t_j)$ volte all'istante $\tau_j + d_j$ E' possibile quindi definire il vettore riga $\mathbf{e}_{ab}$ con n colonne (pari al numero di transizioni) tale che $\mathbf{e}_{ab}(j) = enab(\mathbf{M}, t_j)$.

Esempio 2.5 *In figura 2.2 è mostrata la rete presa in esame per l'esempio. L'insieme dei posti e delle transizioni sono:*

$$P = \{p_1, p_2, p_3\}$$

$$T = \{t_1, t_2, t_3\}.$$

Le matrici $\boldsymbol{Pre}$ *e* $\boldsymbol{Post}$ *sono le seguenti:*

$$\boldsymbol{Pre} = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

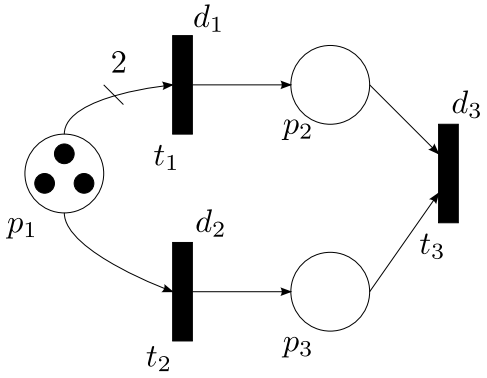


Figura 2.2: Rete di Petri temporizzata.

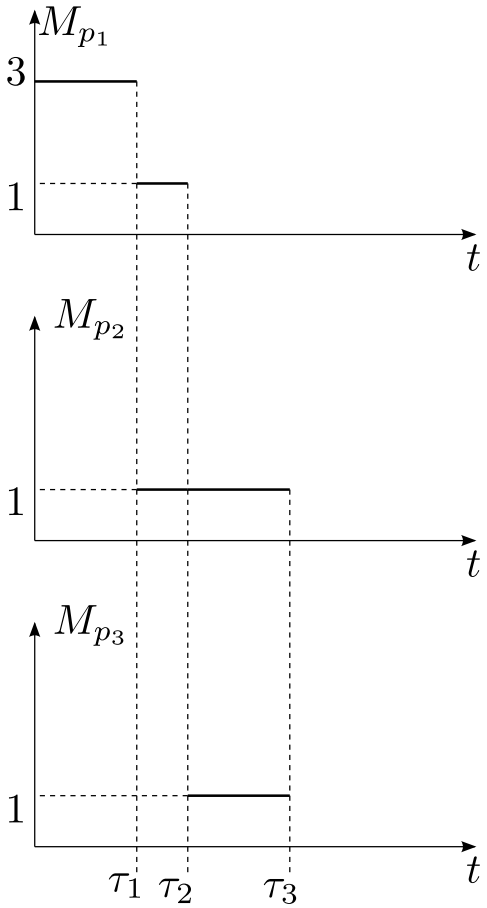


Figura 2.3: Evoluzione Rete di Petri temporizzata in figura 2.2.

$$\boldsymbol{Post} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

La marcatura iniziale è:

$$\boldsymbol{M}_0 = \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix}.$$

Assumiamo che $d_2 > d_1 > 0$ *e che* $d_3 = d_1$*. All'istante di tempo iniziale* $\tau = 0$*,* t_1 *e* t_2 *sono abilitate mentre* t_3 *non lo è. All'istante di tempo* $\tau_1 = d_1$ *la transizione* t_1 *scatta portando la marcatura della rete a* $\boldsymbol{M}_1 = [1\ 1\ 0]^T$*. La transizione* t_1 *non è più abilitata mentre la transizione* t_2 *lo è ancora; quindi all'istante di tempo* $\tau_2 = d_2$ *la transizione* t_2 *scatta e la rete raggiunge la marcatura* $\boldsymbol{M}_2 = [0\ 1\ 1]^T$ *che abilita la transizione* t_3*; infine, all'istante di tempo* $\tau_3 = \tau_2 + d_3$ *la transizione* t_3 *scatta raggiungendo la marcatura* $\boldsymbol{M}_3 = [0\ 0\ 0]^T$*. Ora, nessuna transizione è abilitata. Si noti inoltre che:*

$$enab(\boldsymbol{M}_0, t_1) = 1$$

$$enab(\boldsymbol{M}_0, t_2) = 3.$$

2.1.4 Conflitto

In una qualsiasi rete di Petri N vi è un conflitto strutturale se esiste un posto p_i dal quale escono almeno due transizioni ossia se:

$$\exists p_i \in P : |p_i^\bullet| \geq 2. \tag{2.6}$$

Dove qua e nel seguito si indicherà con $|A|$ la cardinalità dell'insieme A. Un conflitto strutturale viene indicato con la coppia $K = \langle p_i, p_i^\bullet \rangle$. Si noti come la nozione di conflitto strutturale non coinvolga la marcatura della rete.

In una qualsiasi rete di Petri N vi è un conflitto effettivo se esiste un conflitto strutturale K e una marcatura $\mathbf{M}$ 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:

$$\begin{cases} & \exists p_i \in P : \quad |p_i^\bullet| \geq 2 \\ m(p_i) < & \displaystyle\sum_{t_j \in T \wedge e_{ab}(j) > 0} Pre(p_i, t_j). \end{cases} \tag{2.7}$$

Un *conflitto effettivo* è rappresentato attraverso la tripla:

$$K^E = \langle p_i, p_i^\bullet, \mathbf{M} \rangle. \tag{2.8}$$

In una qualsiasi rete di Petri temporizzata vi è un conflitto generale se esiste un conflitto strutturale K e una marcatura $\mathbf{M}$ tale per cui il numero

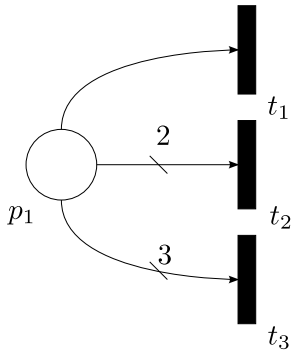


Figura 2.4: Rete di Petri P/T con conflitto strutturale.

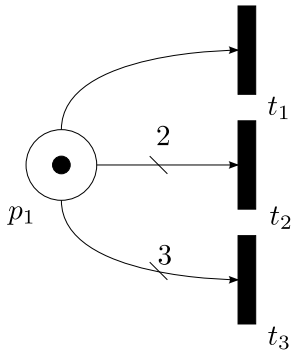


Figura 2.5: Rete di Petri P/T che non presenta conflitto effettivo.

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

$$\left\{ \quad \exists p_i \in P : |p_i^\bullet| \geq 2 \ \mathbf{M}(p_i) < \mathbf{Pre}(p_i, \cdot)\mathbf{e}_{ab}^T. \right. \tag{2.9}$$

Un conflitto generale si rappresenta con la tripla :

$$K^G = \langle p_i, p_i^\bullet, \mathbf{M} \rangle. \tag{2.10}$$

Esempio 2.6 *Si consideri la rete di Petri P/T in figura 2.4; si verifica facilmente che*

$$p_1^\bullet = \{t_1, t_2, t_3\}$$

quindi $|p_1| = 3 \geq 2$ *ed è presente un conflitto strutturale. Ovviamente non è presente conflitto effettivo poichè nessuna transizione è abilitata. Anche la rete in figura 2.5 non presenta conflitto effettivo poiché l'unica transizione abilitata è* t_1*. La rete in figura 2.6 presenta invece conflitto effettivo infatti* $\boldsymbol{e}_{ab} = [2 \ 1 \ 0]^T$ *e quindi dalla seconda delle 2.7 si ottiene* $M_{p_1} = 2 < 3$*. Infine, si verifica facilmente che la rete in figura 2.7, non presenta conflitto effettivo.*

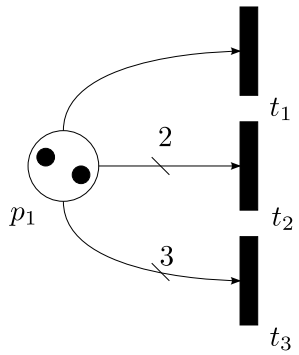


Figura 2.6: Rete di Petri P/T marcata che presenta conflitto effettivo.

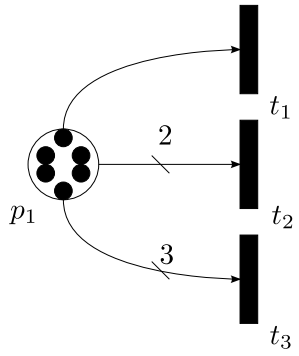


Figura 2.7: Rete di Petri P/T marcata che non presenta conflitto effettivo.

2.2 Reti di Petri Ibride

In questa tesi consideriamo reti di Petri Ibride del primo ordine (*First Order Hybrid Petri Nets*, FOHPN), un modello che consiste di posti continui che mantengono fluido, posti discreti contenenti un numero intero non negativo di gettoni e transizioni anche esse discrete e continue [4].

Assumiamo che una struttura autonoma di temporizzazione (*timing*) sia associata alle transizioni discrete. Al contrario assumiamo che le velocità di scatto istantanee (*Istantaneous Firing Speeds*, IFS) delle transizioni continue possano essere scelte dall'operatore di sistema all'interno di un dato insieme $S \subset (\mathbb{R}_{0^+})^{n_c}$ dove n_c è il numero delle transizioni continue e $\mathbb{R}_{0^+} = \mathbb{R}_+ \cup \{0\}$. Come in tutti i modelli ibridi distinguiamo due livelli di comportamento. A livello più basso, l'evoluzione continua della rete è descritta da modelli di fluido del primo ordine. A livello più alto un sistema ad eventi discreti descrive il macro comportamento della rete che, al verificarsi dei macro eventi, evolve attraverso una sequenza di macro stati. Sessego [15] distingue quattro tipi di macro eventi:

- π_i: un posto continuo si svuota. Questo evento cambia il grado di abilitazione di un insieme di transizioni continue da fortemente abilitate a debolmente abilitate.

- 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.

- ϵ_i: la marcatura di un posto continuo p_i aumenta, raggiungendo un livello di fluido che abilita un insieme di transizioni discrete.

- $\overline{\epsilon_i}$: la marcatura di un posto continuo p_i decresce, raggiungendo un livello di fluido che disabilita un insieme di transizioni discrete.

Si definisce τ_k, per $k = 0, 1, 2 \cdots$, l'istante di tempo in cui si verifica il k-esimo macro evento. L'intervallo di tempo che intercorre tra due macro eventi prende il nome di macro-periodo. Durante tutta la durata del macro-periodo la velocità delle transizioni continue rimane costante. Ad ogni macro periodo:

- si permette che la velocità di scatto istantanea di una transizione continua t_i sia scelta da un agente di controllo in un dato range $[V_i', V_i]$ dove V_i' è la velocità minima di scatto (*minimum firing speed*, mfs) e V_i è la velocità massima di scatto (*maximum firing speed*, MFS);

- si definisce esplicitamente l'insieme di tutti i vettori di velocità di scatto istantanee ammissibili $\mathbf{v}$ in un macro stato. Questo insieme

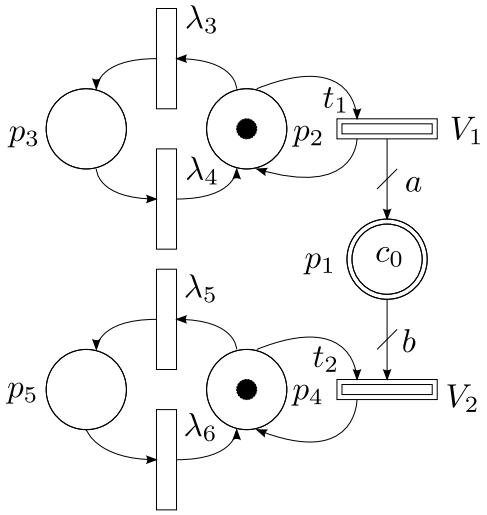


Figura 2.8: Rete di Petri ibrida.

è caratterizzato da delle soluzioni ammissibili di un insieme di vincoli lineari.

- Si considera il problema di scegliere un vettore di velocità di scatto istantanee ottimo in accordo con una funzione obbiettivo data. Questo schema di ottimizzazione sarà miopico, in quanto la soluzione sarà ottima solo nel macro periodo corrispondente.

- Per risolvere il problema di programmazione lineare si utilizza il metodo del simplesso.

2.2.1 Struttura e marcatura

Una rete di Petri ibrida è una struttura $N = (P, T, \mathbf{Pre}, \mathbf{Post}, \mathcal{D}, \mathcal{C})$. L'insieme dei posti $P = P_d \cup P_c$ è partizionato nell'insieme dei posti discreti (rappresentati come cerchi) e nell'insieme di posti continui P_c (rappresentati come doppi cerchi). L'insieme delle *transizioni* $T = T_d \cup T_c$ è partizionato nell'insieme delle transizioni discrete T_d e un insieme delle delle transizioni continue T_c (rappresentate come dei doppi rettangoli). L'insieme $T_d = T_I \cup T_D \cup T_E$ in [15] è così partizionato:

1. T_I è l'insieme delle transizioni *immediate* (rappresentate da delle barre);

2. T_D è l'insieme delle transizioni *temporizzate deterministiche* (rappresentate da rettangoli pieni);

3. T_E è l'insieme delle transizioni *temporizzate esponenzialmente distribuite* (rappresentate da rettangoli vuoti).

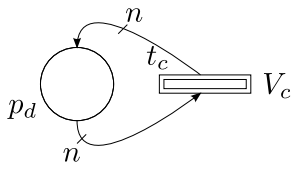


Figura 2.9: Rete di Petri ibrida ben formata.

La cardinalità degli insiemi T, T_d, T_c è indicata rispettivamente con n, n_d, n_c.

Le funzioni di pre-incidenza e post-incidenza che specificano il peso degli archi sono:

$$\mathbf{Pre} : \begin{cases} P_d \times T & \to \mathbb{N} \\ P_c \times T & \to \mathbb{R}_0^+ \end{cases} \tag{2.11}$$

$$\mathbf{Post} : \begin{cases} P_d \times T & \to \mathbb{N} \\ P_c \times T & \to \mathbb{R}_0^+. \end{cases} \tag{2.12}$$

Si richiede, affinchè la rete sia ben formata (*well formed net*), che per ogni $t \in T_c$ e per ogni $p \in P_d$ $Pre(p,t) = Post(p,t)$. In figura 2.9 è mostrato un esempio di rete di Petri ibrida ben formata dove $n \in \mathbb{N}$. La funzione $\mathcal{D} : T_d/T_I \to \mathbb{R}^+$ specifica la temporizzazione associata alle transizioni discrete. Si associa ad una transizione temporizzata deterministica $t_i \in T_D$ un ritardo di scatto costante $d_i = \mathcal{D}(t_i)$. Si associa ad una transizione temporizzata esponenzialmente distribuita $t_i \in T_E$ un tempo medio di scatto $\lambda_i = \mathcal{D}(t_i)$. La funzione $\mathcal{C} : T_c \to \mathbb{R}_0^+ \times \mathbb{R}_\infty^+$ specifica le velocità di scatto associate alle transizioni continue. Per ogni transizione continua $t_i \in T_c$, si pone $\mathcal{C}(t_i) = (V_i', V_i)$, con $V_i' \leq V_i$. Qui V_i' rappresenta la velocità di scatto minima e V_i rappresenta la velocità di scatto massima. Nel, seguito se non specificato altrimenti, la velocità minima di scatto delle transizioni continue sarà $V_i' = 0$. In maniera analoga a quanto visto nelle reti di Petri P/T indichiamo il pre-insieme (post-insieme) delle transizioni t rispettivamente con ${}^\bullet t$, ($t^\bullet$) e le loro restrizioni ai posti continui o discreti con ${}^{(d)}t =^\bullet t \cap P_d$ o ${}^{(c)}t =^\bullet t \cap P_c$. Una notazione analoga è usata per i pre-insiemi e post-insiemi di posti.

La matrice di incidenza della rete è definita, in maniera analoga con quanto visto nelle reti di Petri P/T : $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$. La restrizione di $\mathbf{C}$ a P_X e P_Y ($X, Y \in \{c, d\}$) è indicata con $\mathbf{C}_{XY}$. Si noti che dalle ipotesi di rete ben formata, segue $\mathbf{C}_{dc} = 0$.

Si estende la definizione di marcatura vista nelle reti di Petri P/T alle reti di Petri ibride:

Definizione 2.7 *Una* marcatura *è una funzione che associa a ciascun posto discreto un numero non negativo di gettoni, rappresentati da dei pallini neri*

e assegna a ciascun posto continuo un volume di fluido rappresentato con un numero reale; M_p indica la marcatura del posto p. Il valore di una marcatura all'istante τ è indicato con $\boldsymbol{M}(\tau)$. La restrizione di $\boldsymbol{M}$ alla parte discreta e continua è indicata rispettivamente con $\boldsymbol{M}^d$ e $\boldsymbol{M}^c$, rispettivamente:

$$\boldsymbol{M}: \begin{cases} P_d & \to \mathbb{N} \\ P_c & \to \mathbb{R}_0^+. \end{cases} \tag{2.13}$$

Un *sistema FOHPN* $\langle N, \mathbf{M}(\tau_0) \rangle$ è formato da una rete di Petri ibrida del primo ordine N con una marcatura iniziale $\mathbf{M}(\tau_0)$.

2.2.2 Abilitazione e scatto

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

Definizione 2.8 *Sia$\langle N, \boldsymbol{M} \rangle$ un sistema FOHPN. Una transizione t è* abilitata *dalla marcatura $\boldsymbol{M}$ se per ogni $p \in^\bullet t$, $M_p \geq Pre(p,t)$. Una transizione discreta abilitata t scatta (dopo il relativo ritardo associato) raggiungendo la marcatura $\boldsymbol{M}$'=$\boldsymbol{M}$+$\boldsymbol{C}(\cdot, t)$.*

Una transizione continua è abilitata solo dalla marcatura dei suoi posti discreti in ingresso. Tuttavia, la marcatura dei posti continui in ingresso è usata per distinguere tra abilitazione forte e debole.

Definizione 2.9 *Sia $\langle N, \boldsymbol{M} \rangle$ un sistema FOHPN. Una transizione continua t è* abilitata *dalla marcatura $\boldsymbol{M}$ se per ogni $p \in {}^{(d)}t$, $M_p \geq \boldsymbol{Pre}(p,t)$.*

Definizione 2.10 *Diciamo che una transizione $t \in T_c$ abilitata è:*

- *fortemente abilitata (*strongly enabled*) dalla marcatura $\boldsymbol{M}$ se per tutti i posti $p \in^{(c)} t$, $M_p > 0$;*
- *debolmente abilitata (*weakly enabled*) dalla marcatura $\boldsymbol{M}$ se per qualche $p \in^{(c)} t$, $M_p = 0$.*

Lo stato di abilitazione di una transizione continua t_i definisce la sua velocità di scatto istantanea ammissibile v_i.

Definizione 2.11 *Sia $\langle N, \boldsymbol{M} \rangle$ un sistema FOHPN e $t_i \in T_c$ una transizione continua con velocità di scatto istantanea v_i.*

- *Se t_i non è abilitata allora $v_i = 0$.*
- *Se t_i è fortemente abilitata, allora può scattare con qualsiasi velocità $v_i \in [V_i', V_i]$.*

- *Se t_i è debolmente abilitata, allora può scattare con qualsiasi velocità $v_i \in [V_i', \overline{V_i}]$ dove $\overline{V_i} \leq V_i$ dipende dall'ammontare di fluido entrante nei posti continui vuoti di t_i. Infatti, la transizione non può rimuovere più fluido da ciascun posto vuoto di quanto ne venga immesso dalle altre transizioni.*

La velocità di scatto istantanea all'istante τ di una transizione $t_i \in T_c$ è indicata con $v_i(\tau)$. L'evoluzione nel tempo della marcatura di un posto $p \in P_c$ è descritta da:

$$\frac{dM_p(\tau)}{d\tau} = \sum_{t_i \in T_c} \mathbf{C}(p, t_i) \cdot v_i(\tau). \tag{2.14}$$

Nella 2.14 si assume che all'istante di tempo τ nessuna transizione discreta è scattata e che tutte le velocità $v_i(\tau)$ sono continue in τ.

2.2.3 Dinamica della rete

Come detto in precedenza un macro evento si verifica quando:

1. scatta una transizione discreta;
2. un posto continuo diventa vuoto.

Siano τ_k e τ_{k+1} gli istanti di tempo di macro eventi consecutivi; l'intervallo di tempo $[\tau_k, \tau_{k+1})$ è chiamato macro periodo e la sua durata è indicata con $\Delta_k = \tau_{k+1} - \tau_k$. Si assume che la velocità di scatto istantanea delle transizioni continue sia costante durante il macro periodo. Quindi, la marcatura discreta e il vettore delle velocità di scatto istantanee durante un macro periodo definiscono un macro stato che corrispondono a degli stati invarianti di comportamento.

Si descrive ora la dinamica di una rete di Petri ibrida del primo ordine. Sia τ_0 l'istante iniziale, τ_k $(k > 0)$ gli istanti nei quali si verificano i macro eventi, e $\mathbf{v}(\tau_k)$ il vettore delle velocità di scatto istantanee durante il macro periodo di lunghezza δ_k. Sia $\boldsymbol{\sigma}(\tau_k)$ il vettore contatore di scatti all'istante τ_k. Quindi, il micro-comportamento di una rete di Petri ibrida è descritto durante il k-esimo macro periodo da:

$$\begin{cases} \mathbf{M}^c(\tau) &= \mathbf{M}^c(\tau_k) + \mathbf{C}_{cc} \cdot v(\tau_k) \cdot (\tau - \tau_k) \\ \mathbf{M}^d(\tau) &= \mathbf{M}^d(\tau_k) \end{cases} \tag{2.15}$$

dove $\tau \in [\tau_k, \tau_{k+1})$, mentre l'evoluzione della rete al verificarsi dei macro eventi è descritta da

$$\begin{cases} \mathbf{M}^c(\tau_k) &= \mathbf{M}^c(\tau_k^-) + \mathbf{C}_{cd} \cdot \boldsymbol{\sigma}(\tau_k) \\ \mathbf{M}^d(\tau_k) &= \mathbf{M}^d(\tau_k^-) + \mathbf{C}_{dd} \cdot \boldsymbol{\sigma}(\tau_k). \end{cases} \tag{2.16}$$

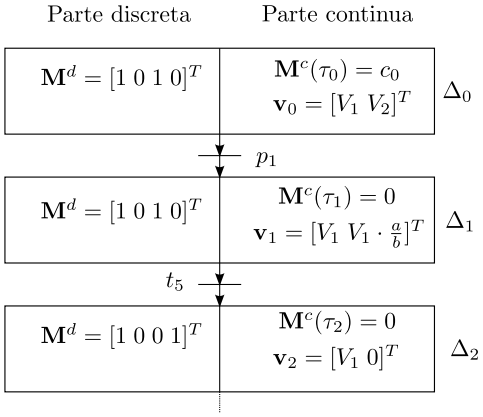


Figura 2.10: Diagramma di fase relativo alla rete 2.8.

Il macro comportamento di una rete di Petri ibrida del primo ordine può essere descritto da un diagramma di fase nel quale ogni macro stato è rappresentato da un rettangolo etichettato sulla destra dalla lunghezza Δ_k del corrispondente macro periodo. Ciascun rettangolo è diviso in due parti. Sulla sinistra (parte discreta) è rappresentata la marcatura discreta $\mathbf{M}^d$. Sulla destra (parte continua) è rappresentata la marcatura continua $\mathbf{M}^c$ della rete all'istante τ_k con il vettore di velocità di scatto istantanee $\mathbf{v}$. I macro stati sono connessi da delle barre, rappresentanti i macro eventi che causano le transizioni di stato. Ciascuna barra è etichettata sulla sinistra (rispettivamente destra) dalle transizioni discrete (rispettivamente, posti continui) che causano il verificarsi dei macro eventi.

Per illustrare meglio questi concetti presentiamo un esempio tratto dalla letteratura [4].

Esempio 2.12 *Si consideri la rete mostrata in figura 2.8 la cui evoluzione è mostrata in 2.11, il relativo diagramma di fase in figura 2.10 e sia* $\tau_0 = 0$ *l'istante di tempo iniziale. Il posto* p_1 *è un posto continuo la cui marcatura iniziale* $M_{p1}(\tau_0) = c_0 > 0$. *I posti* p_2, p_3, p_4, p_5 *sono posti discreti. Le transizioni* t_1 *e* t_2 *sono transizioni continue con velocità massima di scatto pari rispettivamente a* V_1 *e* V_2. *Si assume che* $V_1 \cdot a < V_2 \cdot b$ *(dove* a *e* b *sono i pesi dei rispettivi archi Pre e Post). Le transizioni discrete* t_3, t_4, t_5, t_6 *sono transizioni stocastiche esponenzialmente distribuite i cui tassi di scatto medio sono* λ_3, λ_4, λ_5 *e* λ_6 *Vediamo una possibile evoluzione della rete appena descritta. Formalmente:*

$$P_d = \{p_2, p_3, p_4, p_5\}$$

$$P_c = \{p_1\}$$

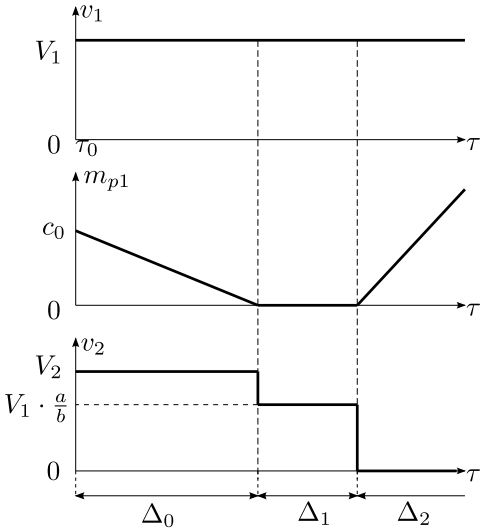


Figura 2.11: Evoluzione della rete in 2.8.

$$\boldsymbol{Pre}_{dd} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{Post}_{dd} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\boldsymbol{Pre}_{cc} = \begin{bmatrix} 0 & b \end{bmatrix}$$

$$\boldsymbol{Post}_{cc} = \begin{bmatrix} a & 0 \end{bmatrix}$$

$$\boldsymbol{Pre}_{cd} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{Post}_{cd} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{Pre}_{dc} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\boldsymbol{Post}_{dc} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

- *Primo macro periodo: inizialmente, p_1 non è vuoto e i posti p_2, p_4 sono marcati. Quindi le transizioni t_1 e t_2 sono fortemente abilitate e possono scattare alla loro velocità massima $v_1 = V_1$ e $v_2 = V_2$. La marcatura continua della rete durante questo macro periodo evolve secondo la seguente equazione:*

$$\boldsymbol{M}^c(\tau) = M_{p1}(\tau) = c_0 - (V_2\, b - V_1\, a)(\tau - \tau_0);$$

la marcatura discreta rimane invece costante e sarà data da :

$$\boldsymbol{M}_0^d = [M_{p2}\ M_{p3}\ M_{p4}\ M_{p5}]^T = [1\ 0\ 1\ 0]^T.$$

- *Secondo macro periodo: all'istante $\tau_1 = (c_0/(V_2 b - V_1 a)) + \tau_0$ il posto p_1 diventa vuoto e questo causa un macro evento. Nel nuovo macro stato, t_1 rimane fortemente abilitata, mentre t_2 è debolmente abilitata e la sua massima velocità di scatto sarà $\overline{V}_2 = v_1(a/b) < V_2$. Ipotizzando $v_1 = V_1$ e $v_2 = V_1(a/b)$. Quindi, la marcatura continua della rete durante questo macro periodo rimarrà costante $M_{p1}(\tau) = 0$. La marcatura discreta mantiene il valore che aveva nel precedente macro periodo non essendo scattata alcuna transizione discreta.*

- *Terzo macro periodo: Si assume che la transizione abilitata t_5 scatti all'istante $\tau_2 > \tau_1$. Questo macro evento cambia la marcatura discreta della rete che diventa:*

$$\boldsymbol{M}_2^d = [1\ 0\ 0\ 1]^T.$$

Ora la transizione t_2 è disabilitata (es. $v_2 = 0$) mentre t_1 rimane fortemente abilitata. Ipotizzando $v_1 = V_1$, la marcatura continua durante questo macro periodo sarà data da:

$$\boldsymbol{M}_c(\tau) = M_{p1}(\tau) = V_1 a(\tau - \tau_2).$$

Il comportamento è mostrato in figura 2.11 che mostra l'evoluzione nel tempo di M_{p1}, v_1, v_2.

2.2.4 Vettori di velocità istantanea di scatto ammissibili

In questa sezione si caratterizza l'insieme dei vettori velocità istantanea di scatto ammissibili.

Definizione 2.13 *Sia $\langle N, \boldsymbol{M} \rangle$ un sistema FOHPN con n_c transizioni continue e matrice di incidenza $\boldsymbol{C}$. Sia $T_\epsilon(\boldsymbol{M}) \subset T_c$ ($T_{\mathcal{N}}(\boldsymbol{M}) \subset T_c$) il sottoinsieme delle transizioni continue abilitate (non abilitate) da $\boldsymbol{M}$, e $P_\epsilon = \{p \in$*

$P_c|M_p = 0\}$ *il sottoinsieme posti continui vuoti. Qualsiasi vettore di velocità istantanea di scatto ammissibile* $v = [v_1 \cdots v_{n_c}]^T$ *nella marcatura* $\boldsymbol{M}$ *è soluzione ammissibile del seguente sistema lineare:*

$$\begin{cases} (a) & V_j - v_j \geq 0 & \forall t_j \in T_\epsilon(\boldsymbol{M}) \\ (b) & v_j - V_j' \geq 0 & \forall t_j \in T_\epsilon(\boldsymbol{M}) \\ (c) & v_j = 0 & \forall t_j \in T_{\mathcal{N}}(\boldsymbol{M}) \\ (d) & \sum_{t_j \in T_\epsilon} \boldsymbol{C}(p, t_j) \cdot v_j \geq 0 & \forall p \in P_\epsilon(\boldsymbol{M}) \end{cases} \tag{2.17}$$

L'insieme di tutte le soluzioni ammissibili è indicato con $\mathcal{S}(N, \mathbf{M})$. Quindi, il numero totale di vincoli che definisce $\mathcal{S}(N, \mathbf{M})$ è : $2 \cdot |T_\epsilon(\mathbf{M})| + |T_{\mathcal{N}}(m)| + |P_\epsilon(\mathbf{M})|$ (qui, $|A|$ indica la cardinalità dell'insieme A).

Osservazione 2.14 *: Ogni vincolo nella forma 2.17d legato ai posti continui p vuoti può essere riscritto come:*

$$\sum_{j \in J} \alpha_j v_j \geq \sum_{k \in K} \alpha_k v_k \tag{2.18}$$

con $J \cap K = \emptyset$ *e* $\alpha_j, \alpha_k \in \mathbb{R}^+$*. L'insieme J (rispettivamente, K) contiene gli indici delle transizioni continue che scattano incrementando (rispettivamente, decrescendo) la marcatura del posto p.*

Definizione 2.15 *: Sia* $\langle N, \boldsymbol{M} \rangle$ *un sistema FOHPN. Una transizione* $t_j \in T_c$ *è chiamata* mfs-free *nella marcatura* $\boldsymbol{M}$ *se almeno una delle seguenti condizioni si verifica:*

1. *la velocità di scatto minima di* t_j *è* $V_j' = 0$*;*
2. t_j *non ha in ingresso posti continui vuoti, es:* $\forall p \in {}^{(c)}t : M_p > 0$.

Condizione sufficiente per l'esistenza di vettori di velocità istantanea di scatto ammissibili:

Proposizione 2.16 *Sia* $\mathcal{S}(N, \boldsymbol{M})$*l'insieme lineare definito in 2.17. Tale insieme è non vuoto se ogni* $t_j \in T_\epsilon(\boldsymbol{M})$ *è* mfs-free *in* $\boldsymbol{M}$*.*

Consideriamo un esempio tratto dalla letteratura [4] per chiarire meglio i concetti esposti.

Esempio 2.17 *Si consideri la rete in figura 2.12. I posti* p_2*,* p_3*,* p_4*,* p_5*,* p_6*,* p_7 *sono posti discreti. I posti* p_1*,* p_8 *sono posti continui. Le transizioni* t_2*,* t_3*,* t_4*,* t_5*,* t_6*,* t_7 *sono transizioni discrete con tempo di scatto esponenzialmente distribuito e pari rispettivamente a* λ_2*,* λ_3*,* λ_4*,* λ_5*,* λ_6*,* λ_7*. Le transizioni* t_1*,* t_2*,* t_9 *sono transizioni continue le cui velocità minime e massime di scatto*

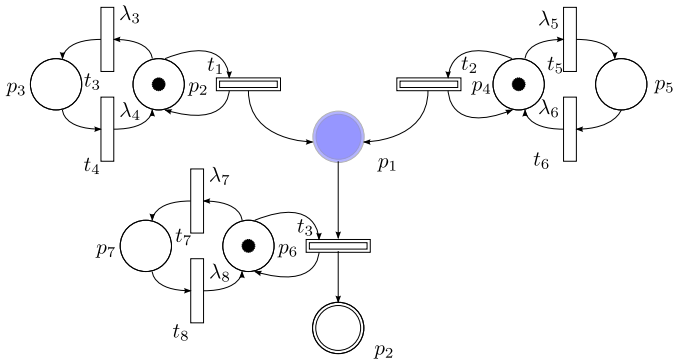


Figura 2.12: Rete di Petri ibrida.

sono indicate tra parentesi quadre. La marcatura discreta della rete è data dal vettore:

$$\boldsymbol{M}^d = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T.$$

La marcatura continua della rete è data dal vettore:

$$\boldsymbol{M}^c = \begin{bmatrix} 0 & 0 \end{bmatrix}^T.$$

L'insieme di ammissibilità è dato dal seguente sistema di equazioni:

$$\mathcal{S}(N, \boldsymbol{M}) = \begin{cases} 5 - v_1 \geq 0 \\ 10 - v_2 \geq 0 \\ 11 - v_9 \geq 0 \\ v_1 - 2 \geq 0 \\ v_2 - 4 \geq 0 \\ v_9 - 3 \geq 0 \\ v_1 + v_2 - v_9 \geq 0. \end{cases}$$

2.2.5 Conflitto

In questa sezione viene definito il concetto di conflitto in una rete di Petri ibrida. Viene trattato il conflitto solo per i posti continui in quanto il calcolo di un vettore di velocità istantanee di scatto ammissibile è influenzato solo da questo tipo di conflitti.

Esempio 2.18 *Si consideri la rete mostrata in 2.13 quando il posto p non è vuoto, sia t_2 che t_3 possono scattare al loro MFS. Quando il posto p è vuoto, tuttavia, il flusso di uscita $v_2 + v_3$ è limitato dal flusso di ingresso v_1, quindi in $\mathcal{S}(N, \boldsymbol{M})$ ci sarà un vincolo nella forma 2.17d legato al posto p tale che $v1 \geq v2 + v3$. Questo vincolo esprime il fatto che si ha un ammontare limitato di risorse (flusso in ingresso) che deve essere condiviso tra differenti processi (transizioni in uscita). Non c'è conflitto in una rete se ciascun posto vuoto $p \in P_c$ ha al massimo una transizione abilitata in uscita $t \in T_c$. Questo motiva la seguente definizione.*

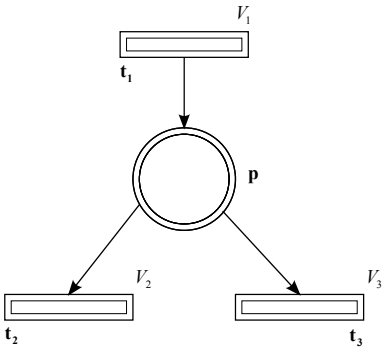


Figura 2.13: Conflitto *free-choice*.

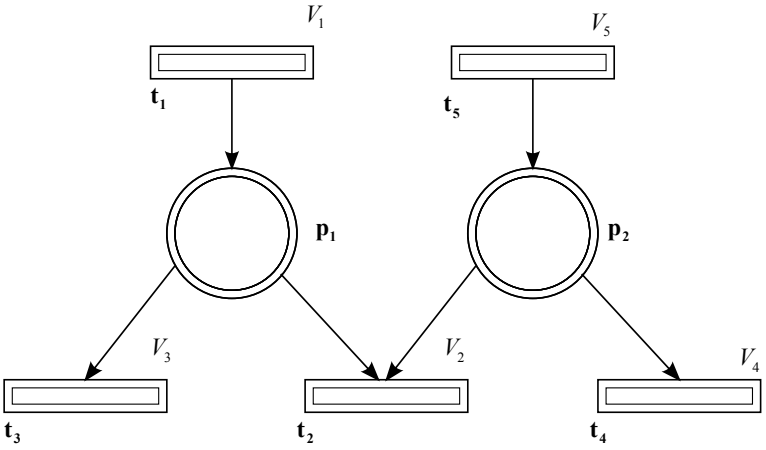


Figura 2.14: Conflitto non *free-choice*.

Definizione 2.19 *Sia* $\langle N, \boldsymbol{M} \rangle$ *un sistema FOHPN e* $\mathcal{S}(N, \boldsymbol{M})$ *l'insieme lineare definito da 2.17. Diciamo che N è* continuous conflict free *(CCF) nella marcatura* $\boldsymbol{M}$ *se per tutti i vincoli della forma 2.17 riscritte nella forma 2.18 si verifica:* $|K| \leq 1$.

Se l'obbiettivo è massimizzare la velocità di scatto delle transizioni continue è possibile mostrare che in una CCF FOHPN ciascuna componente del vettore IFS può essere massimizzata indipendentemente.

Teorema 2.20 *Sia* $\langle N, \boldsymbol{M} \rangle$ *un sistema FOHPN. Se N è CCF nella marcatura* $\boldsymbol{M}$*, la soluzione ottima* $\boldsymbol{v}^*$ *del seguente problema di programmazione lineare (LPP):*

$$\begin{array}{ll} max & \boldsymbol{1}^T \cdot \boldsymbol{v} \\ s.\ t. & \boldsymbol{v} \in \mathcal{S}(N, \boldsymbol{M}) \end{array} \tag{2.19}$$

è tale che $\forall \boldsymbol{v} \in \mathcal{S}(N, \boldsymbol{M}),\ \boldsymbol{v} \leq \boldsymbol{v}^*$.

2.2.6 Risoluzione conflitto globale

Quando la rete non è CCF, non tutte le velocità di scatto possono essere massimizzate indipendentemente. Tuttavia si può sempre trovare una politica di risoluzione del conflitto risolvendo un LPP finalizzata alla ottimizzazione globale delle risorse del sistema. Si possono considerare diversi indici di performance :

- *Massimizzazione flussi* in una rete di Petri ibrida del primo ordine si può considerare come ottima la soluzione $\mathbf{v}^*$ che massimizza l'indice di performance $J = \mathbf{1}^T \cdot \mathbf{v}$ che nel dominio manifatturiero corrisponde a massimizzare l'utilizzo delle macchine.

- *Massimizzazione flusso di uscita* In una rete di Petri ibrida del primo ordine si può massimizzare l'indice di prestazione $J = \mathbf{a}^T \cdot \mathbf{v}$ dove:

$$a_j = \begin{cases} 1, \text{ se } & t_j \quad \text{è esogena} \\ 0, \text{ se } & t_j \quad \text{è endogena} \end{cases} \tag{2.20}$$

  nel dominio manifatturiero questo corrisponde a massimizzare il *throughput*.

- *Bilanciamento flusso* : questo problema consiste nel ridurre la differenza tra l'utilizzo massimo e minimo delle transizioni continue. L'utilizzo di una transizione $t_j \in T_c$ puo essere dato dal rapporto v_j/V_j. Allora si può voler minimizzare l'indice $J = \max_{j \in K}\{v_j/V_j\} - \min_{j \in K}\{v_j/V_j\}$. Nel dominio manifatturiero, questo corrisponde a bilanciare il carico delle macchine.

- *Minimizzare fluido immagazzinato*: In una FOHPN, si può voler minimizzare la derivata della marcatura di un posto $p \in P_c$. Questo può essere fatto minimizzando l'indice di performance $J = \mathbf{a}^T \cdot \mathbf{v}$ dove :

$$a_j = \begin{cases} \mathbf{C}(p, t_j), \text{ se } t_j \in p^{(c)} \cup {}^{(c)}p \\ 0, \text{ altrimenti} \end{cases} \tag{2.21}$$

Una differente procedura di ottimizzazione è basata sulle priorità globali (GP). In questo caso si ha una prestazione multi-obbiettivo nel quale gli obbiettivi hanno diverse priorità. Per prima cosa si osservano tutte le soluzioni che ottimizzano il primo obbiettivo, poi il secondo e così via.

Definizione 2.21 *Sia $\langle N, \boldsymbol{M} \rangle$ un sitema FOHPN e $\mathcal{S}(N, \boldsymbol{M})$ l'insieme lineare definito da 2.17. Si assuma che le transizioni continue della rete siano ordinate in una sequenza di priorità $t_1 > t_2 > t_{nc}$. La soluzione GP-ottima per $\mathcal{S}(N, \boldsymbol{M})$ è definita da:*

$$\begin{array}{ll} v_1^* = & max\{v_1 | \boldsymbol{v} \in \mathcal{S}(N, \boldsymbol{M})\} \\ v_2^* = & max\{v_2 | \boldsymbol{v} \in \mathcal{S}(N, \boldsymbol{M}), v_1 = v_1^*\} \\ v_3^* = & max\{v_3 | \boldsymbol{v} \in \mathcal{S}(N, \boldsymbol{M}), v_1 = v_1^*, v_2 = v_2*\} \\ \cdots & \end{array} \tag{2.22}$$

dove $\boldsymbol{v}^* = [v_1^* \cdots v_{nc}^*]^T$.

La soluzione GP-ottima può essere trovata risolvendo n_c LPP. Inoltre è possibile dimostrare il seguente teorema.

Teorema 2.22 *La soluzione GP-ottima v^* è unica ed è un vertice della regione ammissibile $\mathcal{S}(N, \boldsymbol{M})$*

2.2.7 Risoluzione conflitto locale

L'uso di un indice di performance da massimizzare (o minimizzare) nello spazio dei vettori di velocità istantanee di scatto ammissibili corrisponde ad una procedura di ottimizzazione globale. Capita spesso, tuttavia, che le regole locali siano utilizzate per determinare il modo di operare di un sistema descritto da una rete di Petri ibrida. Si considera il caso delle reti in cui tutti i conflitti sono *free-choice* ossia se un posto continuo p ha più di una transizione in uscita continua (es: $p^{(c)} = \{t_1, t_2, \cdots t_k\}$ con k>1), allora questo è il solo posto continuo in ingresso a tutte le transizioni (es: ${}^{(c)}t_j = p, j = 1, \cdots, k$). Quando i conflitti non sono free-choice, le regole di ottimizzazione locale descritte in seguito non possono essere utilizzate.
Rapporto Fisso: Una particolare semplice regola che può essere utilizzata per risolvere localmente i conflitti *free-choice* è quella di assegnare un rapporto

fisso di fluido di volume per tutte le transizioni continue abilitate che portano fluido fuori da un posto continuo vuoto.

Priorità Locali: Si può considerare anche il caso delle regole di priorità locali attraverso una modifica dell'insieme lineare 2.17. Si supponga che in figura 2.14 una soluzione legale sia tale che t_2 abbia priorità su t_3 (tutto il fluido entrante nel posto p deve essere consumato da t_2 e solo se $v_2 = V_2$ allora il fluido rimanente dovrà essere consumato da t_3). Questo può essere fatto aggiungendo i seguenti vincoli:

$$\begin{cases} Mx \geq V_2 - v_2 \\ v_3 \leq M(1-x) \end{cases} \tag{2.23}$$

dove $x \in \{0,\ 1\}$, $M \in \mathbb{R}$ con $M \gg 0$. Quindi, se $v_2 < V_2$ segue $v_3 = 0$. L' inconveniente di questa tecnica è che un semplice problema di programmazione lineare si è trasformato in un problema più complesso intero-lineare.

Capitolo 3

Il simulatore HYPENS

Il simulatore (sviluppato da Sessego [15]), permette di simulare qualsiasi rete di Petri temporizzata fornitagli in ingresso, che può essere discreta, continua o ibrida. L'acronimo **HYPENS** sta per Hybrid Petri Nets Simulator. Il software è costituito da quattro file

1. enter_HPN.m
2. make_HPN.m
3. simulator_HPN.m
4. analysis_HPN.m.

E' realizzato sia in lingua italiana che in lingua inglese, diversificando quindi la visualizzazione di tutti i commenti a video a seconda della scelta iniziale. Il fulcro del simulatore è il file *simulator_HPN.m* fornisce in uscita l'evoluzione temporale del sistema che viene memorizzata in una struttura dati di output denominata *Evol* grazie alla quale è possibile fare sia delle analisi che delle rappresentazioni grafiche. L'utilizzo dei primi due file è a discrezione dell'utente il quale può decidere o di creare la rete , passo per passo , rispondendo a delle domande visualizzate a schermo , oppure può inserire i dati direttamente nell'input della *function* (secondo file).

3.1 Il linguaggio di programmazione MATLAB

MATLAB è un linguaggio di programmazione matematico orientato alla manipolazione delle matrici e dei vettori. Le istruzioni fondamentali che possono essere usate sono in diretta corrispondenza con delle funzioni che vengono usate normalmente in matematica per il calcolo matriciale e vettoriale 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'utilizzo in funzioni più complesse. Le funzioni complesse di uso più comune sono già disponibili in MATLAB come facenti parte di Toolbox, ovvero librerie dedicate ad applicazioni matematiche specialistiche (Signal Processing, Controllo di Processi, Statistica etc.) Due particolarità di MATLAB rispetto ai linguaggi di programmazione più diffusi sono la possibilità di utilizzarlo con la *line* 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 e, 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.

3.2 Caratteristiche di HYPENS

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**: è 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 *step-by-step*: il software è scritto in Turbo Pascal 6.

- **HISIm**: è 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**: è 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**: è 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**: è 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). I vantaggi principali di HYPENS sono i seguenti:

1. è 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. E' un simulatore di facile utilizzo, anche a livello didattico.

3. E' in grado si simulare indifferentemente reti di Petri temporizzate, reti di Petri continue e reti di Petri ibride.

4. E' in grado di gestire i conflitti tra le transizioni.

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

6. E' in grado di fornire dati statistici sia numericamente che graficamente.

3.3 Funzione enter_HPN

Il file enter_HPN è realizzato attraverso la seguente *function*:
[Pre, Post, M0, vel, v, D, s, alfa] = enter_HPN.
In ingresso non deve essere inserito 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.* Nel seguito si analizzano i dati che vengono chiesti, a partire dal momento in cui viene mandato in esecuzione il file, e i dati che vengono restituiti in uscita.

3.3.1 Dati in ingresso

Nel momento in cui si manda in esecuzione il file enter_HPN.m appare a video la seguente richiesta:

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

Si decide la lingua che deve essere utilizzata durante tutta l'esecuzione del file *enter_ HPN.m*. Se si fornisce un valore diverso da 0 e 1, viene settata automaticamente la lingua inglese. Se si è selezionata la lingua italiana appariranno le seguenti richieste:

- **Inserire il numero di posti continui presenti nella rete**: se nella rete ci sono posti continui, si inserisce il valore che ne indica il numero altrimenti si digita zero.
- **Inserire il numero di posti discreti presenti nella rete**: se nella rete ci sono posti discreti, si inserisce il valore che ne indica il numero altrimenti si digita zero.
- **Inserire il numero di transizioni continue presenti nella rete**: se nella rete ci sono transizioni continue, si inserisce il valore che ne indica il numero altrimenti si digita zero.
- **Inserire il numero di transizioni discrete presenti nella rete**: se nella rete ci sono transizioni discrete, si inserisce il valore che ne indica il numero altrimenti si digita zero.
- **Digitare la matrice di Pre incidenza CC della rete che si vuole considerare**: è una matrice di *pre-incidenza* che riguarda solo la parte continua, che tiene conto di tutti gli archi Pre che collegano un posto continuo $p_c \in P_c$ con una transizione continua $t_c \in T_c$; 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**: è una matrice di *pre-incidenza* che riguarda la parte ibrida, che tiene conto di tutti gli archi Pre che collegano un posto continuo $p_c \in P_c$ con una transizione discreta $t_d \in T_d$; 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 discreto $p_d \in P_d$ con una transizione continua $t_c \in T_c$; 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**: è una matrice di *pre-incidenza* che riguarda la parte discreta, che tiene conto di tutti gli archi Pre che collegano un posto discreto $p_d \in P_d$ con una transizione discreta $t_d \in T_d$; 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**: è una matrice di *post-incidenza* che riguarda solo la parte continua, che tiene conto di tutti gli archi Post che collegano una transizione continua $t_c \in T_c$ con un posto continuo $p_c \in P_c$; 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:** è una matrice di *post-incidenza* che riguarda solo la parte ibrida, che tiene conto di tutti gli archi Post che collegano una transizione discreta $t_d \in T_d$ con un posto continuo $p_c \in P_c$; 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:** è una matrice di *post-incidenza* che riguarda solo la parte discreta, che tiene conto di tutti gli archi Post che collegano una transizione discreta $t_d \in T_d$ con un posto discreto $p_d \in P_d$; 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 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 Pc in un vettore riga tipo** $[\mathbf{M(P1)} \cdots \mathbf{M(Pc)}]$**:**
  si deve inserire un vettore riga indicante la quantità di fluido presente nei posti continui$p_c \in P_c$; se nella rete non ci sono posti continui, tale richiesta non è visualizzata.

- **Inserire la marcatura iniziale dei posti Pd in un vettore riga tipo** $[\mathbf{M(Pc+1)}; \cdots ; \mathbf{M(P)}]$**:** si deve inserire un vettore riga indicante il numero di gettoni presente nei posti continui $p_d \in P_d$; se nella rete $P_d = \emptyset$, tale richiesta non è visualizzata.

In seguito vengono richiesti parametri che riguardano solo la parte discreta che riguardano solamente le transizioni $t_d \in T_d$; quindi, se si sta 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 la caratterizzano.

- **Inserire il vettore riga $[\mathbf{s(Tc+1)};\cdots;\mathbf{s(T)}]$ 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.

3.3.2 Dati restituiti in uscita

I dati che vengono restituiti dal file *enter_ HPN.m* e sono strutturati in modo tale che lo stesso file *simulator_ HPN.m* possa interpretare in modo corretto 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 :**

$$Pre = \begin{pmatrix} Pre_{CC} & NaN & Pre_{CD} \\ NaN & NaN & NaN \\ Pre_{DC} & NaN & Pre_{DD} \end{pmatrix} \tag{3.1}$$

$$Post = \begin{pmatrix} Post_{CC} & NaN & Post_{CD} \\ NaN & NaN & NaN \\ Post_{DC} & NaN & Post_{DD} \end{pmatrix}. \tag{3.2}$$

- **Marcatura iniziale:**

$$M_0 = \begin{bmatrix} M(P1) \\ M(P2) \\ . \\ . \\ . \\ M(P) \end{bmatrix}. \tag{3.3}$$

Questo è il vettore colonna indicante la marcatura iniziale al tempo $\tau = 0$; è costituito dalla concatenazione della marcatura continua con

la marcatura discreta. Se nella rete non ci sono posti continui, il vettore M_0 coinciderà con il vettore della sola marcatura discreta e viceversa se non ho posti discreti nella rete.

- **Velocità delle transizioni continue:**

$$vel = \begin{bmatrix} vel_{min}(T_1) & vel_{max}(T_1) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ vel_{min}(T_c) & vel_max T_c \end{bmatrix}. \tag{3.4}$$

Questa matrice 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.**
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 $T_d = \emptyset$ allora D ha dimensione nulla.

$$D = \begin{bmatrix} param_1(T_c+1) & param_2(T_c+1) & param_3(T_c+1) \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ param_1(T) & param_2(T) & param_3(T) \end{bmatrix}. \tag{3.5}$$

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 $T_d = \emptyset$ allora D ha dimensione nulla.

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

$$v = [v_{T_c+1} \cdots v_T] \tag{3.6}$$

$$s = [s_{T_c+1} \cdots s_T] \tag{3.7}$$

$$alfa = \begin{bmatrix} alfa_{1,T_{c+1}} \cdots alfa_{1,T} \\ alfa_{2,T_{c+1}} \cdots alfa_{2,T} \end{bmatrix}. \tag{3.8}$$

3.4 Funzione make_HPN.m

Il file make_ HPN è realizzato dalla 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 devono essere inseriti i parametri della rete che 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.*

3.4.1 Dati in ingresso

- *Pre_ CC*: matrice degli archi che uniscono un posto continuo $p_c \in P_c$ ad una transizione continua $t_c \in T\_c$ la cui dimensione è $m_c \times n_c$; se nella rete $P_c = \emptyset$ oppure $T_c = \emptyset$ allora la matrice *Pre_ CC* è vuota.

- *Pre_ CD*: matrice degli archi che uniscono un posto continuo $p_c \in P_c$ ad una transizione discreta $t_d \in T_d$ la cui dimensione è $m_c \times n_d$; se nella rete $P_c = \emptyset$ oppure $T_d = \emptyset$ allora la matrice *Pre_ CD* è vuota.

- *Pre_ DC*: matrice degli archi che uniscono un posto discreto $p_d \in P_d$ ad una transizione continua $t_c \in T_c$ la cui dimensione è $m_d \times n_c$; se nella rete $P_d = \emptyset$ oppure $T_c = \emptyset$ allora la matrice *Pre_ DC* è vuota.

- *Pre_ DD*: matrice degli archi che uniscono un posto discreto $p_d \in P_d$ ad una transizione discreta$t_d \in T_d$ la cui dimensione $m_d \times n_d$ se nella rete $P_d = \emptyset$ oppure $T_d = \emptyset$ allora la matrice *Pre_ DD* è vuota.

- *Post_ CC*: matrice degli archi che uniscono una transizione continua $t_c \in T_c$ ad un posto continuo $p_c \in P_c$ la cui dimensione $m_c \times n_c$; se nella rete $P_c = \emptyset$ oppure $T_c = \emptyset$ allora la matrice *Post_ CC* è vuota.

- *Post_ CD*: matrice degli archi che uniscono una transizione discreta $t_d \in T_d$ ad un posto continuo $p_c \in P_c$ la cui dimensione è $m_c \times n_d$; se nella rete $P_c = 0$ oppure $T_{dc} = 0$ allora la matrice *Post_ CD* è vuota.

- *Post_ DC*: matrice degli archi che uniscono una transizione continua $t_c \in T_c$ ad un posto discreto $p_d \in P_d$ la cui dimensione $m_d \times n_c$; se nella rete $P_d = \emptyset$ oppure $Tc = \emptyset$ 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 discreta $t_d \in T_d$ ad un posto discreto $p_d \in P_d$ la cui dimensione $m_d \times n_d$; se nella rete $P_d = \emptyset$ oppure $T_d = \emptyset$ 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 inserito è $1 \times nc$. Se $P_c = \emptyset$ allora il vettore *M0_ P_ c* è vuoto.

- *M0_ Pd*: valore della marcatura iniziale dei posti discreti $p_d \in P_d$ che indica il numero di gettoni contenuti al tempo iniziale $\tau = 0$; la dimensione del vettore inserito è $1 \times m_d$ (per questioni di leggibilità) ma verrà memorizzato come un vettore colonna per uniformità con la teoria; se nella rete $P_d = \emptyset$ allora il vettore $M0\_Pd$ è vuoto.

- *vel*: matrice delle velocità che associa rispettivamente, ad ogni transizione continua $t_c \in T_c$ (righe della matrice), un valore minimo e massimo di velocità (colonne della matrice); la dimensione della matrice è $nc \times 2$. Se nella rete $Tc = 0$ 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 ad ogni transizione T_d; è fortemente legata al tipo di transizione indicata 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 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 ad 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 nella rete è presente un conflitto generale (se scriviamo *alfa* come un vettore possiamo inserire o 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 livello di priorità (livello più basso priorità maggiore). Il numero di righe può essere 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.

3.4.2 Dati restituiti in uscita.

I dati che vengono restituiti in uscita sono analoghi a quelli della funzione *enter_HPN*.

3.5 Funzione simulator_HPN.m

L'interfaccia della *function* è la seguente:
[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 si deve passare l'uscita di una delle due *function* precedentemente trattate e cioè di *enter_HPN.m* oppure *make_HPN.m*.

3.5.1 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 simulazione: *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 ad 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 ed 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 se *it = 0*.

3.5.2 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 all' istante di 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 all' istante di tempo *time_ stop*; se $T_c = \emptyset = 0$ allora *IFS_ Evol* sarà un vettore di dimensioni nulle.

- Posto che genera un macro-evento e il relativo macro-evento:
  P_ macro_ Event , *macro_ Evol*. Memorizzano rispettivamente i valori dei posti continui che generano macro-eventi durante l'evoluzione fino al tempo *time_ stop* e il relativo tipo di macro-evento:

  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 macro-evento 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 all' istante di tempo *time_ stop*. Se $T_d = 0$, *firing_ transition* sarà un vettore vuoto mentre se $Td > 0$ e il macro-evento è generato in seguito all' evoluzione di un posto continuo, *firing_ transition* avrà un valore pari a *NaN*.

- Tempo di macro-evento: *timer_ macro_ event*.
  Identifica il tempo associato al macro-evento che si è verificato; se non si verifica nessun macro-evento allora *timer_ macro_ event* avrà un valore pari a *NaN*.

- Durata totale della simulazione: *tau*.
  Identifica il tempo di simulazione totale a partire dall'istante temporale 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 : *Pc_ Pd_ Tc_ Td*. E' un vettore che memorizza i valori di P_c, P_d, T_c e Td.

3.6 Funzione analysis.m

Il file analysis_ HPN è realizzato da 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).
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*, è necessario 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*.

3.6.1 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 si pone *statistic_ plot = 1*, vengono visualizzati due istogrammi in due 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 se si sta simulando una rete di Petri discreta: se si pone *graph = 1*, viene visualizzato il grafo di evoluzione della Rete di Petri temporizzata, dove ad 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 viene rappresentato 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_y, p_z, oppure di fare il grafico di tutte le marcature dei posti ponendo *marking plot = -1*; in quest'ultimo caso è possibile anche creare un vettore *marking_plot* contenente gli indici di tutti i posti della rete, ma per semplicità e leggibilità è possibile ottenere lo stesso risultato imponendo *marking plot = -1*.

- Grafico degli scatti delle transizioni. Se si pone *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 è rappresentato il tempo nell'asse delle ascisse e il valore delle marcature nell'asse delle ordinate differenziate, a seconda del posto, da un colore diverso riconducibile al posto stesso grazie

alla presenza di una finestra che 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_y 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 è possibile ottenere lo stesso risultato imponendo *up_ marking_ plot = -1*.

- Grafico delle probabilità di avere x gettoni nei posti: *Pd_ probability_ plot*. Se poniamo *Pd_ probability_ plot = 1*, vengono visualizzate, in diverse finestre tante quante $|P_d|$, le probabilità di avere x gettoni nel posto $p_d \in P_d$ fino all' istante di tempo massimo $\tau = time\_stop$ che coincide con la fine della simulazione. Nel grafico è rappresentato il numero di gettoni nell' asse delle ascisse e la probabilità corrispondente al numero di gettoni nell' asse delle ordinate.

- Grafico del valore 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 valore medio dei gettoni nei posti fino al tempo $\tau = timestop$. Nel grafico rappresentiamo il tempo nell' asse delle ascisse e il valore medio della marcatura di un posto nell' asse delle ordinate. Dato $x(pi; j) = MarkingAsymptote(pi; j)$, cioè il valore medio della marcatura del posto p_i all' istante τ_j in cui si è verificato l'evento η_j , l'evoluzione nel tempo del valore medio della marcatura del posto p_i è calcolata come:

$$x(p_i, j) = \frac{(x(i, j-1) \cdot \tau(j) + (\tau(j+1) - \tau(j)) \cdot M(i, y(j)))}{\tau(j+1)} \quad (3.9)$$

  $\forall j = \Delta(k), k = 0, 1, 2, \cdots$ E' possibile 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_y e p_z, oppure di fare il grafico dell'evoluzione del valore medio della marcatura in tutti i posti ponendo *Marking_ mean_ Asymptote = -1*; in quest'ultimo caso è ovviamente possibile in alternativa creare un vettore *Marking_ mean_ Asymptote* contenente gli indici di tutti i posti della rete.

- Grafico della velocità media delle transizioni continue: *mean_ speed_ plot*. Se si pone *mean_ speed_ plot = 1*, viene visualizzato, in un unico grafico, il valore medio delle transizioni continue fino al tempo massimo $\tau = time\_stop$. Nel grafico sono rappresentate le transizioni

continue nell'asse delle ascisse e il valore medio della velocità associato alle transizioni stesse nell'asse delle ordinate.

- Grafico della frequenza di scatto delle transizioni: *Transition_frequency_plot*. Se si pone *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 sono rappresentate le transizioni discrete nell'asse delle ascisse e le rispettive frequenze di scatto nell'asse delle ordinate.

- Scelta della lingua: *it* Se si pone *it = 0* vine impostata la lingua inglese, ponendo invece $it = 1$ imposto la lingua italiana. Di default è impostato *it = 1*.

3.6.2 Dati in uscita

- *P_ave*: vettore riga di dimensione pari al numero di posti presente nella rete che tiene traccia della marcatura media durante l'esecuzione della simulazione.

- *P_max*: vettore riga di dimensione pari al numero dei posti presente nella rete che tiene traccia del massimo della marcatura durante l'esecuzione della simulazione.

- *Pd_ave_t*: matrice che ha un numero di righe pari a quello dei posti discreti presenti nella rete, numero di colonne pari al numero di macro-eventi che si verificano durante l'esecuzione della simulazione. La generica colonna k specifica la marcatura media dei posti discreti dall'istante $t_0 = 0$ all'istante t_k in cui si verifica il k-esimo macro-evento.

- *IFS_ave*: vettore riga di dimensione pari al numero di transizioni continue che tiene traccia della velocità media delle transizioni continue.

- *Td_ave*: vettore riga di dimensione pari al numero di transizioni discrete che tiene traccia del tempo di abilitazione delle transizioni discrete.

- *Pd_freq*: matrice con numero di righe pari a quello dei posti discreti e numero di colonne pari al massimo numero di gettoni (che viene raggiunto durante la simulazione) in ogni posto discreto più uno .

- *Md_freq*: matrice con numero di righe pari a quello dei posti discreti più uno e numero di colonne pari a quello delle differenti marcature discrete che vengono raggiunte durante l'esecuzione della simulazione. Ciascuna colonna di *Md_freq* una di queste marcature e nell'ultima riga la corrispondente frequenza.

3.7 Estensione del simulatore ed esempio applicativo

In questa sezione si presenteranno le estensioni apportate al simulatore al fine di renderlo utilizzabile nell'ambito dell'applicazione a tecniche di analisi perturbativa infinitesimale e le relative simulazioni effettuate per verificarne il corretto funzionamento. Si è modificata la *function simulator_HPN* al fine di estenderne l'utilizzo.

3.7.1 simulator1_HPN

Sono stati aggiunti alcuni parametri in ingresso che consentono, data una marcatura, di:

- modificare l'insieme dei vettori IFS ammissibili $\mathcal{S}(N, \boldsymbol{M})$;
- modificare i pesi della funzione obbiettivo.

Si ricorda infatti che ad ogni macro-evento, al fine di determinare il vettore delle velocità istantanee ottimo, occorre risolvere il PPL:

$$\begin{cases} \max & \mathbf{c}^T \cdot \mathbf{v} \\ \mathbf{v} \in & \mathcal{S}(N, \boldsymbol{M}) \end{cases} \tag{3.10}$$

dove il vettore $\mathbf{c}$ non varia da macro-periodo a macro-periodo e l'insieme $\mathcal{S}(N, \boldsymbol{M})$ è così definito:

$$\mathcal{S}(N, \boldsymbol{M}) = \begin{cases} (a) & V_j - v_j \geq 0 & \forall t_j \in T_\epsilon(\boldsymbol{M}) \\ (b) & v_j - V_j' \geq 0 & \forall t_j \in T_\epsilon(\boldsymbol{M}) \\ (c) & v_j = 0 & \forall t_j \in T_{\mathcal{N}}(\boldsymbol{M}) \\ (d) & \sum_{t_j \in T_\epsilon} C(p, t_j) \cdot v_j \geq 0 & \forall p \in P_\epsilon(\boldsymbol{M}). \end{cases} \tag{3.11}$$

L'interfaccia della function è la seguente:
[Type,M_Evol,IFS_Evol,P_macro,Event_macro_Evol,
firing_transition,timer_macro_event,
tau,Q_Evol,timer,nc_nd_qc_qd]=
simulator1_HPN(Pre,Post,M,vel,v,D,
s,alpha,time_stop,simulation_type,it,c_m,ab_m,theta,cont_places_const,c).
Si può osservare che i parametri in uscita sono gli stessi della *function simulator_HPN* descritta nel capitolo 3 mentre, per quanto riguarda i parametri in ingresso, sono stati aggiunti i seguenti:

- *c_m*: se è pari ad uno allora è possibile specificare in un *m-file* a parte come scegliere ad ogni macro-evento il vettore $\boldsymbol{c}$ in funzione della marcatura raggiunta all'inizio del macro-stato. Se invece *c_m* è pari a zero allora il vettore $\mathbf{c}$ viene mantenuto costante per tutta l'evoluzione della rete.

- *ab_ m*: se è pari ad uno allora è possibile specificare in un *m-file* a parte dei vincoli aggiuntivi che dovranno essere soddisfatti dal vettore delle velocità istantanee. Ad esempio sarà possibile specificare in forma chiusa le velocità assunte dalle transizioni continue in funzione della marcatura della rete.

- *cont_ places_ const*: è un vettore che specifica i posti continui la cui marcatura si vuole vincolare a rimanere costante. Questo risulterà particolarmente utile nella simulazione di quei sistemi che, per esigenze di modellizzazione necessitano di alcuni posti la cui marcatura rimanga costante nel tempo ad esempio in [13] per esigenze di modellizzazione di un impianto di imbottigliamento vengono introdotti dei posti la cui marcatura è vincolata a rimanere nulla.

- *theta*: è un vettore (di dimensione non specificata) contenente dei parametri relativi alla rete ad esempio può rappresentare la dimensione di un *buffer* oppure essere un parametro di *routing*, etc.

Il nuovo PPL da risolvere potrà quindi assumere, nel caso più generale, la seguente forma (nel caso si ponga *c_ m* e *ab_ m* pari ad uno):

$$\begin{cases} \max & \mathbf{c}^T(\mathbf{M}) \cdot \mathbf{x} \\ \text{s.t.} & \\ \mathbf{x} \in & \mathcal{S}(N, \mathbf{M}) \cap \mathcal{A}(N, \mathbf{M}); \end{cases} \tag{3.12}$$

dove $\mathcal{A}(N, \mathbf{M})$ è il nuovo insieme di vincoli che dovranno essere specificati dall'utente. I vincoli aggiuntivi devono essere specificati dall'utente negli *m-file f_ ab_ m.m* e *f_ c_ m.m*.

3.7.2 Simulazione impianto di imbottigliamento

In questa sezione vogliamo mostrare, tramite un esempio di simulazione, come, grazie alle modifiche apportate ad *HYPENS*, si possano simulare reti in cui, per esigenze di modellizzazione, sia necessario vincolare la marcatura di qualche posto continuo a rimanere costante nel tempo o in particolare nulla. Sarà sufficiente specificare in un vettore i posti continui la cui marcatura si vuole vincolare a rimanere costante e il simulatore genererà automaticamente in fase di esecuzione gli opportuni vincoli. La schematizzazione dell'impianto di imbottigliamento è mostrata in figura 3.1 e in figura 3.2 è mostrata la modellizzazione tramite reti di Petri ibride tratta dalla letteratura [13]. Confrontando le due figure è facile convincersi che la macchina M_1 è rappresentata dalla transizione t_1, la macchina M_2 è rappresentata dalla transizione continua t_2, la macchina assemblatrice M_3 è rappresentata dalla transizione continua t_3, il *buffer* è rappresentato dal posto continuo p_2 e il posto continuo p_1 modellizza un elemento connettore (*pipe*). Il resto della

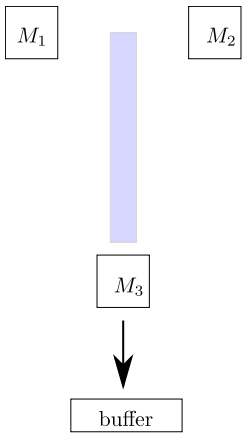


Figura 3.1: Schematizzazione dell'impianto di imbottigliamento.

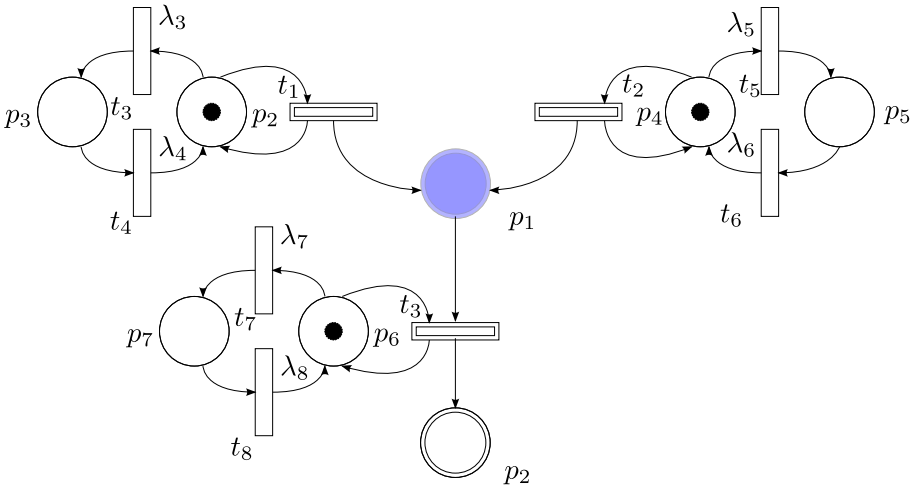


Figura 3.2: Rete che modellizza l'impianto di imbottigliamento.

rete (parte discreta) modellizza lo stato di guasto o non guasto delle macchine. In questo tipo di modello l'insieme dei posti continui $P_c = P_0 \cup P_+$ viene partizionato nell'insieme dei posti continui P_0 (su sfondo blu) la cui marcatura deve essere vincolata a rimanere nulla (elementi connettori), e l'insieme dei posti continui P_+ la cui marcatura può assumere qualsiasi valore non-negativo. Si vuole che la marcatura del posto p_1 rimanga costante quindi si porrà:

- cont_places_const = [1].

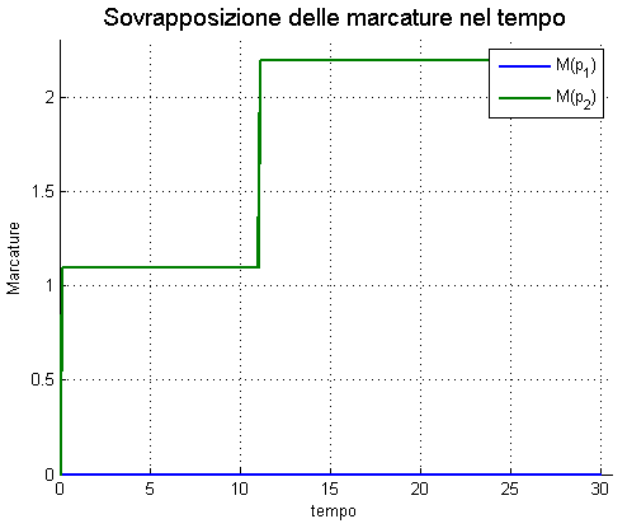


Figura 3.3: Evoluzione della marcatura continua.

Per ulteriori dettagli riguardo i dati di questa simulazione si rimanda all'appendice dove è riportato l'intero codice. L'evoluzione della marcatura, mostrata in figura 3.3 conferma quanto si era voluto imporre.

Capitolo 4

IPA per reti di Petri ibride

4.1 Infinitesimal Perturbation Analysis

L'Analisi perturbativa infinitesimale (*Infinitesimal Perturbation Analysis*, IPA) è una tecnica utilizzata per l'analisi della sensitività dei sistemi ad eventi discreti (DEDS) [9]. Essa fornisce degli algoritmi per la stima dei gradienti delle funzioni di prestazione. Negli anni ottanta e novanta, [16] l'IPA è stata utilizzata principalmente nei modelli di code, dove le misure di prestazione di ritardo e *throughput* sono di primario interesse. Questo approccio si distingue per la semplicità delle formule che legano i gradienti alle loro funzioni di prestazione. Tuttavia questo approccio iniziale è stato limitato dalla scoperta che in molti modelli realistici di coda i gradienti risultanti presentano un offset statistico, ponendo in dubbio l'utilizzabilità dell'IPA nell'ottimizzazione [11]. Per superare questo problema, l'IPA è stata investigata nelle code fluide (*fluid queues*), che prendono il nome di modelli fluidi stocastici (*Stochastic Flows Models*, SFM) [5].

Ad esempio un semplice ed elegante risultato dell'IPA nello studio dei modelli di fluido stocastici riguarda la perdita media in una coda [5] come funzione della dimensione del *buffer*. In figura 4.1 è schematizzato l'arrivo dei pacchetti nel *buffer*; i pacchetti possono essere accumulati nel buffer finché non si raggiunge la soglia C, a quel punto vengono persi. La derivata di questa funzione di prestazione in un intervallo di tempo lungo T secondi si dimostra essere $-N_T/T$ dove N_T è il numero dei periodi occupati (*busy*)

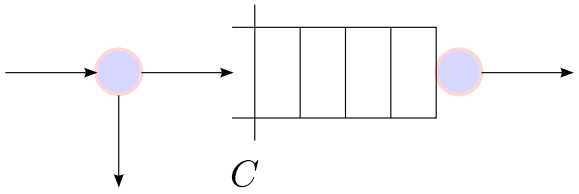


Figura 4.1: Schematizzazione del processo di arrivo dei pacchetti nel buffer.

nell'intervallo $[0, T]$ nei quali si verifica almeno un *overflow*. Un periodo occupato comincia quando il *buffer* non è più vuoto e termina quando il *buffer* torna ad essere vuoto per la prima volta. In figura 4.2 è mostrato un

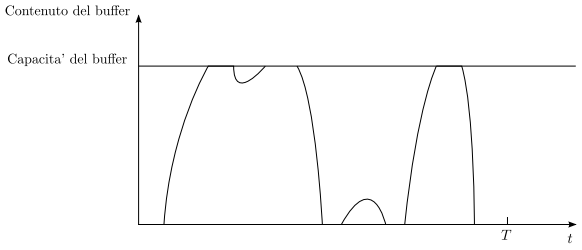


Figura 4.2: Contenuto del buffer.

esempio di traiettoria del contenuto del *buffer*. Dove $N_T = 2$, poiché ci sono tre periodi occupati ma solamente in due di essi vi è perdita.

Il risultato sopra menzionato ha stimolato un ulteriore studio dell' IPA applicata alle reti [9]. Le misure di prestazione di interesse includono flusso utile (*throughput*), perdita (*loss*), ritardo (*delay*) e sono viste come funzioni di variabili come i tassi di servizio (*service rates*), il flusso in ingresso (*inflow rate*) e vari parametri di controllo del flusso. Sono state considerate molte aree di applicazione tra cui le telecomunicazioni [8], i sistemi manifatturieri [2] e il controllo del traffico [3]. In tutti questi studi i gradienti IPA sono ottenuti tramite formule e algoritmi facilmente calcolabili e indipendenti dal modello (*model-free*).

In questo capitolo vengono presentati due esempi fisici in cui il modello delle reti di Petri ibride e l'ottimizzazione tramite tecniche di analisi perturbativa infinitesimale trovano una applicazione naturale. Nella prima sezione si presenterà il modello di una rete manifatturiera con tre macchine e verranno esposte le formule per la stima del gradiente della funzione obbiettivo [2]. Nella seconda sezione si estenderanno alcune formule presentate nella sezione precedente al caso di n macchine.

Nel capitolo successivo invece si simuleranno queste reti grazie alle modifiche apportate ad HYPENS.

4.2 Sensitività del flusso utile nelle transizioni continue

Si consideri lo scenario tipico mostrato in figura 4.3. La transizione continua t_x, chiamata transizione di processo (*processing transition*), ha n posti in ingresso. Ciascun posto in ingresso ha una sorgente di fluido che è rappresentata da una transizione continua, chiamata transizione in ingresso. Sia $v_i(t)$ il tasso di flusso istantaneo dalla transizione di ingresso t_i e $V_x(t)$ il massimo tasso di flusso della transizione di processo, dove $t \in [0, T]$ per un

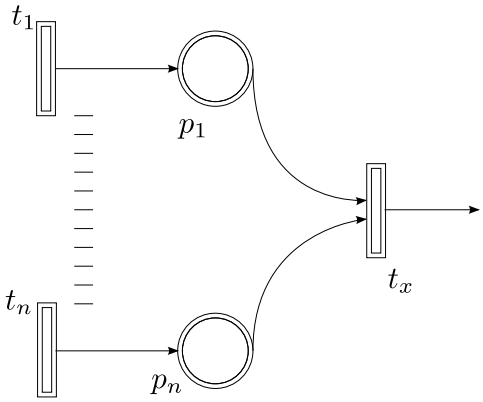


Figura 4.3: Transizioni e posti continui.

dato $T > 0$. Si indica con $M_{p_i}(t)$ l'ammontare di fluido nel posto p_i e con $v_x(t)$ il tasso di flusso istantaneo della transizione di processo. In accordo con la definizione assiomatica dei sistemi dinamici [10] è possibile considerare $v_i(t)$ e $V_x(t)$ come l'ingresso del sistema mentre $M_{p_i}(t),\ i = 1, 2, \cdots n$ e $v_x(t)$ comprendono le sue variabili di stato. Questi processi stocastici sono definiti in uno spazio comune di probabilità $(\Omega, \mathcal{F}, \mathcal{P})$ e si assume che le loro realizzazioni siano continue a tratti. Si definisce l'insieme dei posti vuoti all'istante t:

$$I(t) = \{i = 1, \cdots n \ : M_{p_i}(t) = 0\}. \tag{4.1}$$

Le variabili di stato sono legate ai processi in ingresso dalle seguenti due equazioni:

$$\frac{dM_{p_i}(t)}{dt} = v_i(t) - v_x(t); \tag{4.2}$$

$$v_x(t) = \begin{cases} V_x(t), & \text{se } I(t) = \emptyset \\ \min\{v_i(t),\ i \in I(t)\} & \text{se } I(t) \neq \emptyset. \end{cases} \tag{4.3}$$

Sono necessarie alcune osservazioni.

1. L'equazione 4.3 implica che la transizione di processo scatti alla sua velocità massima finché nessun posto in ingresso è vuoto, mentre un posto in ingresso vuoto rallenterà la velocità di scatto.

2. L'equazione 4.2 richiede che siano date le condizioni iniziali $M_{p_i}(0),\ i = 1, \cdots, n$.

3. Se $I(t)$ non cambia il suo valore in un qualche intervallo aperto $(\overline{\tau}, \overline{t}) \subset [0, T]$, allora $\forall t \in (\overline{\tau}, \overline{t})$ e $\forall i, j \in I(t)$, $v_i(t) = v_j(t)$. In questo caso,

$$v_x(t) = v_i(t),\ \forall t \in (\overline{\tau}, \overline{t}),\ \forall i \in I(t). \tag{4.4}$$

Sia θ un parametro dal quale questo processo dipende. Per esempio, θ può essere il massimo tasso di flusso della transizione di processo, o un parametro della legge di probabilità di qualcuno dei processi in ingresso. Per mantenere la generalità, si assume che tutti i processi in ingresso siano funzioni di θ, e quindi si denotano le loro realizzazioni con $v_i(\theta;t)$ e $V_x(\theta;t)$ rispettivamente. Di conseguenza, tutti gli altri processi sono anch'essi in funzione di θ e verranno indicati con $M_{p_i}(\theta;t)$ e $v_x(\theta,t)$. Si assume che θ sia confinato in un intervallo compatto Θ. La funzione obbiettivo che si desidera massimizzare è il flusso in uscita medio:

$$J(\theta) = \frac{1}{T}\int_0^T v_x(\theta;t)dt. \tag{4.5}$$

Si definisce periodo vuoto (*empty period*) del posto p_i la massima lunghezza positiva del sotto-intervallo $[0,T]$ durante il quale $M_{p_i}(t) = 0$. Si fanno le seguenti ipotesi per ogni $\theta \in \Theta$ fissato.

Ipotesi 4.1 *Con probabilità 1 le funzioni $v_i(\theta;.)$ sono Lipschitziana* [1] *continue a tratti nell'intervallo $t \in [0,T]$.*

Ipotesi 4.2 *Tutti i termini derivativi che saranno menzionati nel seguito esistono con probabilità 1.*

Ipotesi 4.3 *Per ogni $i = 1, \cdots, n$ $\frac{dM_{p_i}}{d\theta}(\theta;0) = 0$.*

E' necessaria qualche osservazione. L'ipotesi 4.1 significa che le funzioni in ingresso sono Lipsichitziane continue a tratti nell'intervallo dove loro sono continue. Questa ipotesi implica che tutti i termini definiti nelle equazioni 4.2, 4.3, 4.4 sono ben definiti. L'ipotesi 4.3 implica che le condizioni iniziali $M_{p_i}(\theta,0)$ sono indipendenti da θ. Si fa una ulteriore ipotesi riguardo gli istanti di partenza e quelli terminali dei periodi vuoti. Le discontinuità nei processi di ingresso $\{v_i(\theta;.)\}$ e $\{V_x(\theta;.)\}$ sono dovuti all'occorrenza di certi eventi discreti. Oltre a questi si considerano come eventi discreti il cambiamento di stato di un posto continuo da vuoto a non vuoto e viceversa.

In maniera analoga a quanto fatto in [5] si classificano questi eventi come segue.

1. Gli eventi esogeni sono delle discontinuità nei processi di ingresso $\{v_i(\theta;.)\}$ e $\{V_x(\theta;.)\}$ la cui temporizzazione è indipendente da θ.

2. Gli eventi endogeni sono l'inizio dei periodi vuoti nei vari posti. Questi eventi sono chiamati macro eventi in [4].

[1] Una funzione $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$ si dice Lipschitziana su Ω se $\exists K \geq 0$ tale che $||f(x) - f(y)|| \leq K||x - y|| \forall x, y \in \Omega$.

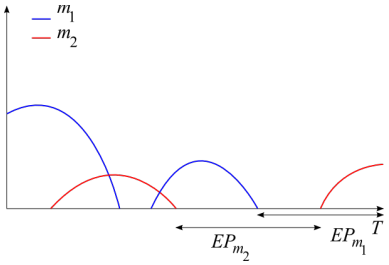


Figura 4.4: Traiettorie di livello nei posti.

3. Un evento indotto è la fine di un periodo vuoto in uno dei posti che è causato (*triggered*) dall'inizio di un periodo vuoto in un altro posto. Infatti, un evento endogeno, può causare un brusco decremento in $v_x(\theta;.)$ e questo può causare la fine di un periodo vuoto in un altro posto, quindi un evento indotto.

Formalmente si fa questa ulteriore ipotesi.

Ipotesi 4.4 *Fissato $\theta \in \Theta$ valgono, con probabilità 1, le seguenti proposizioni:*

1. *Tutte le discontinuità nei processi di ingresso $\{v_i(\theta;.)\}$ e $\{V_x(\theta;.)\}$ sono eventi esogeni.*

2. *Solo un singolo evento esogeno o evento endogeno può verificarsi in un istante di tempo.*

3. *Indicando con $t_{i,j}$ la fine del j-esimo periodo vuoto nel posto p_i, se $t_{i,j}$ non è l'istante di tempo di un evento indotto allora o $\frac{dt_{i,j}}{d\theta} = 0$, oppure la funzione $v_i(\theta,.) - v_x(\theta,.)$ è continua nell'stante $t = t_{i,j}$.*

Corollario 4.5 ([2]) *Se $t_{i,j}$ non è l'istante di tempo di un evento indotto, allora o $\frac{dt_{i,j}}{d\theta} = 0$, oppure la funzione $v_i(\theta,.) - v_x(\theta,.)$ è continua all'istante $t = t_{i,j}$ e $v_i(t_{i,j}) - v_x(t_{i,j}) = 0$.*

Si definisce t_{max} l'ultimo istante di tempo in cui termina in un qualsiasi posto un periodo vuoto. Allora si possono avere due casi: o $t_{max} = T$ oppure $t_{max} < T$. Nel primo caso, mostrato in figura 4.4, l'istante di tempo finale T è contenuto in un periodo vuoto in almeno uno dei posti, e quindi $I(T) \neq \emptyset$ (in figura 4.4 $I(T) = \{1\}$). Vale il seguente risultato:

Proposizione 4.6 ([2]) *Fissato $\theta \in \Theta$*

1. *se $t_{max} = T$, allora per ogni $i \in I(t_{max})$:*

$$\frac{dJ}{d\theta}(\theta) = T^{-1} \int_0^T \frac{\partial v_i}{\partial \theta}(\theta, t)dt; \tag{4.6}$$

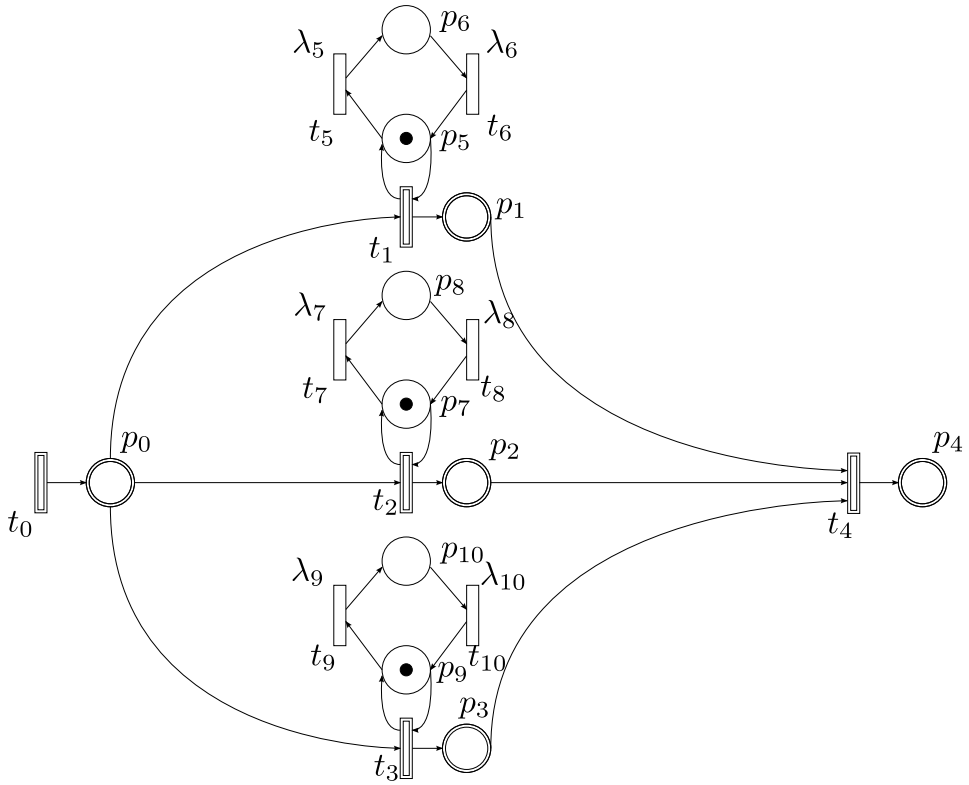


Figura 4.5: Rete manifatturiera con tre macchine.

2. se $t_{max} < T$*, allora per ogni* $i \in I(t_{max})$

$$\frac{dJ}{d\theta}(\theta) = T^{-1}(\int_0^{t_{max}} \frac{\partial v_i}{\partial \theta}(\theta, t)dt) + \int_{tmax}^{T} \frac{\partial V_x}{\partial \theta}(\theta, t)dt). \qquad (4.7)$$

4.3 Modello rete manifatturiera con tre macchine

In figura 4.5 è mostrato il modello con rete di Petri ibrida del sistema manifatturiero preso in esame. Il materiale da lavorare viene immesso nel sistema ad una velocità unitaria dalla transizione continua t_0 e depositato in un *buffer* rappresentato dal posto continuo p_0; viene lavorato parallelamente in tre macchine, rappresentate dalle transizioni continue t_1, t_2 e t_3 e immagazzinato nei *buffers* rappresentati dai posti continui p_1, p_2 e p_3 infine il materiale viene assemblato dalla macchina che è rappresentata dalla transizione continua t_4 e depositato nel *buffer* rappresentato dal posto continuo p_4. Le transizioni t_5, t_6, t_7, t_8, t_9, t_{10} hanno tempi di di scatto esponenzialmente distribuiti e valgono rispettivamente λ_5, λ_6, λ_7, λ_8, λ_9, λ_{10}. La marcatura dei posti p_5, p_6, p_7, p_8, p_9, p_{10} rappresenta lo stato discreto della rete e rappresenta fisicamente se la macchina corrispondente è guasta oppure no. Sia

θ_i, $i = 1, 2, 3$ un parametro a valore reale dal quale il processo dipende. Esso è un parametro di *routing*, rappresenta fisicamente la quantità di fluido che viene indirizzato dal posto p_0 alla transizione t_i quando tutte le macchine sono operative, ossia quando:

$$\mathbf{M}^d = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T.$$

Esso prende il nome di *nominal routing fraction* di t_i. I vincoli che tale parametro deve rispettare sono i seguenti:

- $\theta_i \in [0,1]\ \forall i \in \{1, 2, 3\}$
- $\theta_1 + \theta_2 + \theta_3 = 1$.

Quindi se tutte le macchine sono operative allora:

$$v_i = \theta_i,\ i = 1,\ 2,\ 3. \tag{4.8}$$

La politica che si assume nel caso che qualche macchina ma non tutte sia non operativa è di dividere il flusso tra le macchine operative in proporzione al loro *nominal routing fraction*. Dunque se le macchine i, j sono operative mentre la macchina k è guasta allora:

$$v_i = \theta_i + \frac{\theta_k \theta_i}{\theta_i + \theta_j}; \tag{4.9}$$

e

$$v_j = \theta_j + \frac{\theta_k \theta_j}{\theta_i + \theta_j}. \tag{4.10}$$

Se solo la macchina i è operativa mentre le macchine j e k sono guaste allora

$$v_i = 1; \tag{4.11}$$

mentre

$$v_j = v_k = 0. \tag{4.12}$$

Infine, se tutte le macchine sono guaste allora

$$v_i = 0,\ \ i = 1,\ 2,\ 3. \tag{4.13}$$

In tutti i casi, tranne l'ultimo è facile osservare che $v_1 + v_2 + v_3 = 1$.

L'obbiettivo è determinare i valori di $\theta_1,\ \theta_2$ e θ_3 che massimizzino il flusso utile del sistema. Poiché θ_3 può essere ottenuto da θ_1 e θ_2 si può adottare come parametro di ottimizzazione $\boldsymbol{\theta} = [\theta_1\ \theta_2]^T$. La funzione di prestazione $J(\boldsymbol{\theta})$ è così definita:

$$J(\boldsymbol{\theta}) = T^{-1} \int_0^T v_4(\boldsymbol{\theta}; t) dt \tag{4.14}$$

e il problema di ottimizzazione che si considera è di massimizzare il valore atteso di questa funzione $E[J(\boldsymbol{\theta})]$ dove $E[.]$ indica il valore atteso (*expectation*) in $(\Omega, \mathcal{F}, \mathcal{P})$.

Per l'ottimizzazione di $J(\boldsymbol{\theta})$ viene utilizzato l'algoritmo del tipo Robbins-Monro [7] che calcola una sequenza iterativa $\boldsymbol{\theta}_k := \begin{bmatrix} \theta_{1k} & \theta_{2k} \end{bmatrix}^T \in \mathbb{R}^2, k = 1, 2....$

Algoritmo 4.7 *: Dati:*

- $\boldsymbol{\theta}_1 = [\theta_{11}\ \theta_{21}]^T \in \mathbb{R}^2$ *tale che* $\theta_{i1} \in [0,\ 1]$ *per* $i = 1,\ 2$*;*
- *un* $\epsilon > 0$ *piccolo;*
- *una sequenza positiva* $\{\lambda_k\}_{k=1}^{\infty}$ *che soddisfi le condizioni di convergenza per gli algoritmi di Robbins-Monro, ossia:*
  1. $\sum_{k=1}^{\infty} \lambda_k = \infty$*;*
  2. $\sum_{k=1}^{\infty} \lambda_k^2 < \infty$.

1. *Si ponga* $k = 1$.

2. *Si simuli il sistema per* T *secondi e si calcolino le derivate* $\frac{\partial J(\boldsymbol{\theta_k})}{\partial \theta_i}$, $i = 1,\ 2$.

3. *Per* $i = 1,\ 2,$ *se* $\theta_{ki} + \lambda_k \frac{\partial J(\theta_k)}{\partial \theta_i} \in (\epsilon, 1 - \epsilon)$ *si ponga:*

$$\theta_{i(k+1)} = \theta_{ik} + \lambda_k \frac{\partial J(\theta_k)}{\partial \theta_i} \tag{4.15}$$

   altrimenti porre $\theta_{i(k+1)} = \theta_k$.

4. *Si ponga* $k = k + 1$ *e si torni al* primo passo.

E' necessaria qualche osservazione:

1. il *terzo passo* assicura che $\boldsymbol{\theta}_k$ rimanga ammissibile per ogni $k = 1,\ 2...$ ossia $\boldsymbol{\theta}_k \in [0,\ 1] \times [0,\ 1]$.
2. La convergenza di simili algoritmi in un minimo locale è stata dimostrata [7].
3. I termini in derivate parziali $\frac{\partial v_j}{\partial \theta_i}$ dipendono dalle condizioni di abilitazione delle macchine $M_j,\ j = 1,\ 2,\ 3$ ossia dalla marcatura discreta della rete.

In tabella 4.1 sono mostrate le stime di $\frac{\partial v_i}{\partial \theta}$ in funzione del parametro θ e dello stato di abilitazione delle macchine da utilizzarsi nelle formule della proposizione 4.6 per ottenere una stima del gradiente della funzione di prestazione.

	$\frac{\partial v_1}{\partial \theta_1}$	$\frac{\partial v_1}{\partial \theta_2}$	$\frac{\partial v_2}{\partial \theta_1}$	$\frac{\partial v_2}{\partial \theta_2}$	$\frac{\partial v_3}{\partial \theta_1}$	$\frac{\partial v_3}{\partial \theta_2}$
M_1, M_2, M_3 ON	1	0	0	1	-1	-1
M_1, M_2 ON M_3 OFF	$\frac{\theta_2}{(\theta_1+\theta_2)^2}$	$-\frac{\theta_1}{(\theta_1+\theta_2)^2}$	$-\frac{\theta_2}{(\theta_1+\theta_2)^2}$	$\frac{\theta_1}{(\theta_1+\theta_2)^2}$	0	0
M_1, M_3 ON M_2 OFF	$\frac{1}{1-\theta_2}$	$\frac{\theta_1}{(1-\theta_2)^2}$	0	0	$-\frac{1}{1-\theta_2}$	$-\frac{\theta_1}{(1-\theta_2)^2}$
M_2, M_3 ON M_1 OFF	0	0	$\frac{\theta_2}{(1-\theta_1)^2}$	$\frac{1}{1-\theta_1}$	$-\frac{\theta_2}{(1-\theta_1)^2}$	$-\frac{1}{1-\theta_1}$
2 o più m. OFF	0	0	0	0	0	0

Tabella 4.1: I valori di $\frac{\partial v_i}{\partial \theta_j}$, $i = 1, 2, 3$, $j = 1, 2$ per il caso relativo alla rete manifatturiera con tre macchine.

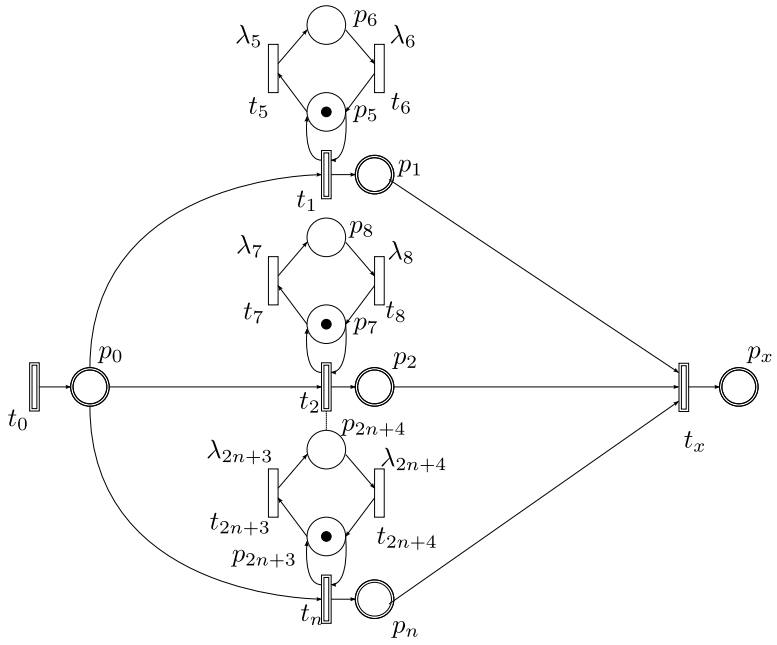


Figura 4.6: Rete manifatturiera con n macchine.

4.4 Rete manifatturiera con n macchine

In questa sezione si intende estendere ad un numero arbitrario di n macchine il modello discusso nella sezione precedente. In figura 4.6 è mostrata la generalizzazione della rete. Si indica con:

- $\mathcal{M} = \{t_1, t_2, \cdots, t_n\}$ l'insieme delle transizioni continue rappresentanti le macchine (non sono comprese ne la macchina che assembla e ne la macchina che immette fluido nella rete).
- $\mathcal{G} \subseteq \mathcal{M}$ l'insieme delle macchine guaste, ossia le macchine che non sono abilitate dalla marcatura discreta. Formalmente:

$$\mathcal{G} = \{t \in \mathcal{M} | \ \exists \ p_d \in \ {}^{(d)}t \ \wedge M_{p_d} = 0\}. \tag{4.16}$$

- $\overline{\mathcal{G}} \subseteq \mathcal{M}$ l'insieme delle macchine non guaste.
- θ_i è il *nominal routing fraction* della macchina associata alla transizione $t_i \in \mathcal{M}$. Ossia è la velocità che la transizione t_i assume quando $\mathcal{G} = \emptyset$.

Ovviamente $\mathcal{G} \cup \overline{\mathcal{G}} = \mathcal{M}$ e $\mathcal{G} \cap \overline{\mathcal{G}} = \emptyset$. I vincoli che le variabili θ_i devono soddisfare sono i seguenti:

- $\theta_i \in [0,\, 1]\ \forall i = 1,\ \cdots n$
- $\sum_{i=1}^{n} \theta_i = 1.$

Per quanto riguarda la velocità delle macchine valgono le seguenti:

1. se $\mathcal{G} = \emptyset$ allora $v_i = \theta_i\ \forall i \in \{i | t_i \in \mathcal{M}\}$;
2. se $\mathcal{G} \neq \emptyset$ e $|\overline{\mathcal{G}}| \geq 2$ allora per ogni $i \in \{i | t_i \in \mathcal{M}\}$ vale:

$$v_i = \begin{cases} 0 & \text{se } i \in \{i | t_i \in \mathcal{G}\} \\ \theta_i + \dfrac{\theta_i \cdot \sum\limits_{k \in \{k | t_k \in \mathcal{G}\}} \theta_k}{\sum\limits_{k \in \{k | t_k \in \overline{\mathcal{G}}\}} \theta_k} & \text{altrimenti} \end{cases} ; \tag{4.17}$$

3. se $\mathcal{G} \neq \emptyset$ e $|\overline{\mathcal{G}}| = 1$ allora per ogni $t_i \in \mathcal{M}$:

$$v_i = \begin{cases} 0 & \text{se } i \in \{i | t_i \in \mathcal{G}\} \\ 1 & \text{altrimenti} \end{cases} ; \tag{4.18}$$

4. se $\mathcal{G} \neq \emptyset$ e $|\overline{\mathcal{G}}| = 0$ allora per ogni $i \in \{i | t_i \in \mathcal{M}\}$ $v_i = 0$.

Si noti che nei casi 1, 2 e 3 è rispettato il vincolo $\sum_{i=1}^{n} v_i = 1$. Nelle tabelle 4.2, 4.3 è riportata la generalizzazione delle formule mostrate in tabella 4.1.

4.5 IPA per l'ottimizzazione delle dimensioni di un buffer

Una delle applicazioni relativamente più semplici dell'IPA che si trova in letteratura [5] è relativa al controllo *on-line* delle dimensioni di un buffer. Il *buffer* accetta pacchetti fino alla quantità di soglia C. In questo caso la funzione obbiettivo da minimizzare è:

$$J(C) = \overline{Q}(C) + R \cdot \overline{L}(C). \tag{4.19}$$

	$dv_i/d\theta_l$
	$i = l \wedge i \neq n$
$\mathcal{G} = \emptyset$	1
	$l = i \wedge l \in \{l \vert t_l \in \overline{\mathcal{G}}\}$
$\mathcal{G} \neq \emptyset \wedge \vert\mathcal{G}\vert < \vert\mathcal{M}\vert - 1 \wedge \mathcal{G} \cap \{t_n\} = \emptyset \wedge i \neq n$	$1 + \frac{\sum_{k \in \{\vert t_k \in \mathcal{G}\}} \theta_k}{1 - \sum_{k \in \{k \vert t_k \in \mathcal{G}\}/\{n\}} \theta_k}$
	$l = i$
$\mathcal{G} \neq \emptyset \wedge \vert\mathcal{G}\vert < \vert\mathcal{M}\vert - 1 \wedge \mathcal{G} \cap \{t_n\} \neq \emptyset \wedge i \neq n$	$1 + \frac{(1 - \sum_{k \in \{k \vert t_k \in \mathcal{G}\}/\{n,i\}} \theta_k) - 2 \cdot \theta_i) \cdot (\sum_{k \in \{k \vert t_k \in \overline{\mathcal{G}}\}} \theta_{k}) - \theta_i \cdot (1 - \sum_{k \in \{k \vert k \in \mathcal{G}\}/\{n\}} \theta_k)}{(\sum_{k \in \{k \vert k \in \overline{\mathcal{G}}\}} \theta_k)^2}$
	$l = i$
$\mathcal{G} \neq \emptyset \wedge \vert\mathcal{G}\vert < \vert\mathcal{M}\vert - 1 \wedge \mathcal{G} \cap \{t_n\} = \emptyset \wedge i \neq n$	0
$\mathcal{G} \neq \emptyset \wedge \vert\mathcal{G}\vert < \vert\mathcal{M}\vert - 1 \wedge \mathcal{G} \cap \{t_n\} \neq \emptyset \wedge i \neq n$	0
$\vert\mathcal{G}\vert \geq \vert\mathcal{M}\vert - 1$	0

Tabella 4.2: I valori di $\frac{\partial v_i}{\partial \theta_j}$, $i = 1, 2, \cdots n$, $j = 1, 2, \cdots n-1$ nel caso generale.

	$dv_i/d\theta_l$
$l = n$	altr.
$\mathcal{G} = \emptyset$ -1	0
$l \neq i \wedge l \in \{l \vert t_l \in \mathcal{G}\}/\{n\}$	
$\mathcal{G} \neq \emptyset \wedge \vert\mathcal{G}\vert < \vert\mathcal{M}\vert - 1 \wedge \mathcal{G} \cap \{t_n\} = \emptyset \wedge i \neq n$	$\frac{\theta_i}{(1-\sum_{k\in\{k\vert t_k\in\mathcal{G}\}/\{n\}}\theta_k)^2}$
	$l \neq i \wedge l \in \overline{\mathcal{G}}$
$\mathcal{G} \neq \emptyset \wedge \vert\mathcal{G}\vert < \vert\mathcal{M}\vert - 1 \wedge \mathcal{G} \cap \{t_n\} \neq \emptyset \wedge i \neq n$	$\frac{-\theta_i}{(\sum_{k\in\{k\vert k\in\overline{\mathcal{G}}\}}\theta_k)^2}$
	$l \neq i \wedge l \in \mathcal{G}$
$\mathcal{G} \neq \emptyset \wedge \vert\mathcal{G}\vert < \vert\mathcal{M}\vert - 1 \wedge \mathcal{G} \cap \{t_n\} = \emptyset \wedge i \neq n$	$-1 + \frac{(-\sum_{k\in\{k\vert t_k\in\mathcal{G}\}}\theta_k + (1-\sum_{k\in\{k\vert t_k\in\mathcal{M}\}/\{n\}}\theta_k))\cdot(1-\sum_{k\in\{k\vert t_k\in\mathcal{G}\}/\{n\}}\theta_k) + (1-\sum_{k\in\{k\vert t_k\in\mathcal{M}\}/\{n\}}\theta_k)\cdot(\sum_{k\in\{k\vert t_k\in\mathcal{G}\}}\theta_k)}{(1-\sum_{k\in\{k\vert t_k\in\mathcal{G}\}/\{n\}}\theta_k)^2}$
$\mathcal{G} \neq \emptyset \wedge \vert\mathcal{G}\vert < \vert\mathcal{M}\vert - 1 \wedge \mathcal{G} \cap \{t_n\} \neq \emptyset \wedge i \neq n$	
$\vert\mathcal{G}\vert \geq \vert\mathcal{M}\vert - 1$	

Tabella 4.3: I valori di $\frac{\partial v_i}{\partial \theta_j}$, $i = 1, 2, \cdots n$, $j = 1, 2, \cdots n-1$ nel caso generale.

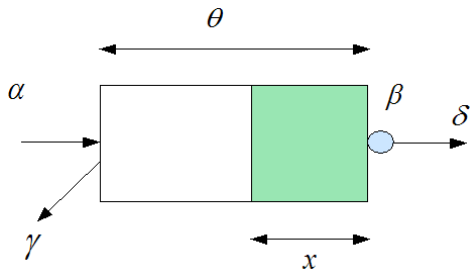


Figura 4.7: Coda

dove $\overline{L}(C)$ è il tasso di perdita atteso (*expected loss rate*), ossia il numero medio di pacchetti persi per un buffer di dimensione C, $\overline{Q}(C)$ è la lunghezza media della coda (*mean queue lenght*), R è un coefficiente di peso e $C \in \mathbb{N}$. Ovviamente, a parità di statistica di sorgente, ad aumentare delle dimensioni del buffer diminuirà il tasso di perdita ma aumenterà la lunghezza media della coda. Quindi $J(C)$ rappresenta il compromesso tra il soddisfacimento del servizio (basso ritardo) e la perdita di troppi pacchetti. Poiché le decisioni di controllo devono essere prese periodicamente una misura di prestazione più realistica è quella in cui $\overline{L(C)}$ e $\overline{Q(C)}$ sono sostituiti da $\overline{L_T(C)}$ e $\overline{Q_T(C)}$ ossia il tasso di perdita medio e la lunghezza media della coda nell'intervallo di tempo $[0, T]$. Si considera dunque:

$$J_T(C) = \overline{Q}_T(C) + R \cdot \overline{L}_T(C). \tag{4.20}$$

Utilizzando un modello fluido la funzione obbiettivo diventa:

$$J_T^{SFM}(\theta) = \overline{Q}_T^{SFM}(\theta) + R \cdot \overline{L}_T^{SFM}(\theta) \tag{4.21}$$

dove $\theta \in \mathbb{R}^+$.

Il processo di arrivo dei pacchetti è modellato come una sorgente ON-OFF; durante il periodo ON i pacchetti arrivano con un tasso α mentre durante il periodo OFF non arriva alcun pacchetto. Per il controllo delle dimensioni del *buffer* è necessario avere una stima di $\frac{dJ_T(\theta)}{d\theta}$ e questa stima deve essere basata sulla osservazione diretta dei dati di una simulazione. E' possibile in questo modo ottenere un θ^* che minimizzi $J_T(\theta)$ attraverso uno schema iterativo del tipo:

$$\theta_{n+1} = \theta_n - \mu_n H_n(\theta_n, \omega_n^{SFM}), \quad n = 0, 1, \cdots \tag{4.22}$$

Dove $\{\nu_n\}$ è una sequenza di passi, $H_n(\theta_n, \omega_n^{SFM})$ è una stima del gradiente della funzione di prestazione $(J_T/(\theta))$ valutata in $\theta = \theta_n$ e ottenuto da una simulazione indicata con ω_n^{SFM}. I processi stocastici in $(\Omega, \mathcal{F}, \mathcal{P})$ sono i seguenti

- $\{\alpha(t)\}$ è la statistica della sorgente, ossia il flusso in ingresso (*inflow rate*).

- $\{\beta(t)\}$ è il massimo rate di scarico della rete dal server, ossia il tasso di servizio (*service rate*).
- $\{\delta(t)\}$ è lo scarico attuale dal server, ossia il flusso in uscita (*output flow*).
- $\{x(t)\}$ volume di fluido nel *buffer*.
- $\{\gamma(t)\}$ è l' *overflow*, ossia il flusso di pacchetti persi dovuto all' ulteriore ingresso di fluido nel *buffer* pieno.

Questi processi evolvono nell' intervallo di tempo [0,T] per un fissato $T > 0$. Si assume che i processi $\{\alpha(t)\}$ e $\{\beta(t)\}$ siano superiormente continui e costanti a tratti(*right-continous piecewise constants*), con $0 \leq \alpha_{min} \leq \alpha(t) \leq \alpha_{max} < \infty$. Il parametro θ indica la dimensione del *buffer* ed è il parametro su cui ci si concentra per l'analisi perturbativa infinitesimale. I processi $\{\alpha(t)\}$ e $\{\beta(t)\}$ sono detti processi definiti (*defined process*) e, insieme alla dimensione del *buffer* θ, definiscono il comportamento del modello di fluido stocastico. In particolare, essi definiscono il contenuto del buffer $x(\theta;t)$, il tasso di *overflow* $\gamma(\theta;t)$ e il flusso di uscita $\delta(\theta;\, t)$. La dipendenza da θ dei processi indica che le misure di prestazione verranno analizzate in funzione di un dato θ. Si assume che un parametro a valore reale θ sia confinato in un intervallo chiuso e limitato (compatto) Θ; per evitare inutili complicazioni si assume che $\theta > 0$ per ogni $\theta > 0$ per ogni $\theta \in \Theta$. Il contenuto del *buffer* $x(\theta; t)$ è determinato dalla seguente equazione differenziale:

$$\frac{dx}{dt} = \begin{cases} 0, & \text{se } x(\theta;t) = 0 \ \wedge \ \alpha(t) - \beta(t) \leq 0 \\ 0, & \text{se } x(\theta;t) = \theta \ \wedge \ \alpha(t) - \beta(t) > 0 \\ \alpha(t) - \beta(t), & \text{altrimenti.} \end{cases} \tag{4.23}$$

Con condizione iniziale $x(\theta, 0) = x_0$; per semplicità si porrà, se non indicato diversamente, $x_0 = 0$. Il tasso di flusso in uscita $\delta(\theta;\, t)$ è dato da:

$$\delta(\theta;\, t) = \begin{cases} \beta(t), & \text{se } x(\theta;\, t) > 0 \\ \alpha(t), & \text{se } x(\theta;\, t) = 0. \end{cases} \tag{4.24}$$

Si evidenzia il fatto che se si permette $\theta = 0$ allora $\delta(\theta;\, t) = \min\{\alpha(t), \beta(t)\}$. Il tasso di *overflow* $\gamma(\theta;\, t)$ è dato da:

$$\gamma(\theta;\, t) = \begin{cases} \max\{\alpha(t) - \beta(t), 0\} & \text{se } x(\theta;\, t) = \theta \\ 0 & \text{se } x(\theta;\, t) < \theta. \end{cases} \tag{4.25}$$

Questo modello di fluido stocastico può essere visto come un sistema dinamico i cui ingressi consistono in due processi definiti e insieme alla dimensione del *buffer* il suo stato comprende il contenuto del buffer, e le sue uscite includono il flusso in uscita e i processi di *overflow*. Lo stato e i processi di uscita sono detti processi derivati, poiché essi sono determinati dai processi

definiti. Poiché le realizzazioni di $\{\alpha(t)\}$ e $\{\beta(t)\}$ sono costanti a tratti e superiormente continue (*piecewise constant and right-continuous*), la traiettoria di stato è lineare a tratti e continua in t, e la funzione di uscita $\gamma(\theta;t)$ è costante a tratti.

Inoltre la traiettoria di stato può essere decomposta in due tipi di intervalli:

1. periodi vuoti;
2. periodi occupati.

I periodi vuoti (EPs) sono i massimi intervalli di tempo durante i quali il *buffer* è vuoto, mentre i periodi occupati (BPs) sono i massimi intervalli di tempo durante i quali il *buffer* non è vuoto. Si osservi che durante i periodi vuoti, il sistema non è necessariamente inattivo; si veda 4.23. Si noti inoltre che poiché $x(\theta;t)$ è continuo in t, i periodi vuoti sono sempre intervalli chiusi, mentre i periodi occupati sono intervalli aperti a meno che contengano uno dei punti terminali 0 o T.

Si considerano due misure di prestazione: il volume perso (*loss volume*) $L_T(\theta)$ e il carico cumulativo (*cumulative workload*) $Q_T(\theta)$ entrambi definiti nell'intervallo di tempo $[0,T]$ dalle seguenti equazioni:

$$L_T(\theta) = \int_0^T \gamma(\theta,t)dt \tag{4.26}$$

$$Q_T(\theta) = \int_0^T x(\theta,t)dt. \tag{4.27}$$

Dove, come già detto, si assume $x(\theta,0)=0$. Si osservi che $(1/T)E[L_T(\theta,t)]$ è il tasso di perdita atteso (*expected loss rate*) nell'intervallo $[0,T]$. In maniera simile $(1/T)E[Q_T(\theta,t)]$ è il contenuto atteso del *buffer* (*expected buffer content*) nell'intervallo $[0,T]$. E' possibile formulare il problema di ottimizzazione come la determinazione di un θ^* tale che minimizzi una funzione di costo nella seguente forma:

$$J(\theta) = \frac{1}{T}E[Q_T(\theta)] + \frac{R}{T}E[L_T(\theta)] = \frac{1}{T}J_Q(\theta) + \frac{R}{T}J_L(\theta). \tag{4.28}$$

Dove R rappresenta un costo di rigetto (*rejection cost*) dovuto all'*overflow*. Per minimizzare la funzione di costo occorre avere una stima di $dJ_L(\theta)/d\theta$ e di $dJ_Q(\theta)/d\theta$. Nella prossima sezione si presenteranno tali stime.

4.6 Stima del gradiente delle funzioni obbiettivo

Si assume che i processi $\{\alpha(t)\}$ e $\{\beta(t)\}$ siano indipendenti da θ e dal contenuto del *buffer*. La stima delle derivate tramite la tecnica dell'analisi perturbativa infinitesimale, calcola $L'_T(\theta)$ e $Q'_T(\theta)$ attraverso un test campione

osservato (*sample path observed*) ω. Una stima basata sull' IPA $\mathcal{L}'(\theta)$ di una misura derivativa di prestazione $dE[\mathcal{L}(\theta)]/d\theta$ è detto imparziale (*unbiased*) se

$$\frac{dE[\mathcal{L}(\theta)]}{d\theta} = E[\frac{d[\mathcal{L}(\theta)]}{d\theta}]. \tag{4.29}$$

L'imparzialità è la principale condizione per fare in modo che la tecnica dell' IPA sia utilizzabile nella pratica. Si consideri una evoluzione campione del modello di fluido stocastico nell'intervallo di tempo $[0, T]$. Per un fissato $\theta \in \Theta$, l'intervallo $[0, T]$ è suddiviso alternativamente in periodi vuoti e periodi occupati. Si supponga che ci siano K periodi occupati indicati con $\mathcal{B}_k$, $k = 1, \cdots K$ in ordine crescente. Le funzioni di prestazione assumono dunque la seguente forma:

$$L_T(\theta) = \sum_{k=1}^{K} \int_{\mathcal{B}_k} \gamma(\theta; t)dt \tag{4.30}$$

$$Q_T(\theta) = \sum_{k=1}^{K} \int_{\mathcal{B}_k} x(\theta; t)dt. \tag{4.31}$$

Come detto precedentemente si assume che i processi $\{\alpha(t)\}$ e $\{\beta(t)\}$ siano costanti a tratti. Si indica il $k - esimo$ periodo occupato con $\mathcal{B}_k = (\zeta_k, \eta_k(\theta))$, $k = 1, \cdots, K$ dove ζ_k è l'istante di tempo in cui il buffer comincia a riempirsi mentre η_k è l'istante di tempo in cui il buffer si svuota. Si può notare come l'istante di tempo in cui il buffer comincia a riempirsi non dipenda dalle dimensioni del buffer in quanto legato ad un evento esogeno, mentre l'istante in cui si svuota si. I periodi occupati inoltre possono essere classificati in base al fatto che in essi avvenga un qualche *overflow* oppure no. Si definiscono inoltre l'insieme dei periodi occupati in cui avviene almeno un overflow :

$$\Phi(\theta) = \{k \in \{1, \cdots K\} : x(t) = \theta \wedge \alpha(t) - \beta(t) > 0 \text{ per qualche } t \in (\zeta_k, \eta_k)\}. \tag{4.32}$$

Per ogni $k \in \Phi(\theta)$, c'è un numero casuale $M_k \geq 0$ di periodi di *overflow* in $\mathcal{B}_k$, ossia intervalli di tempo in cui il *buffer* è pieno e $\alpha(t) - \beta(t) > 0$. Questi periodi di *overflow* vengono indicati con $\mathcal{F}_{k,m}$, $m = 1, \cdots M_k$ e vengono espressi come: $\mathcal{F}_{k,m} = [u_{k,m}(\theta), v_{k,m}]$, $k = 1, \cdots K$. Si osservi che l'istante di partenza dipende da θ mentre, l'istante terminale, è localmente indipendente da θ. Il numero di BPs in $[0, T]$ durante il quale si è osservato qualche *overflow* sarà dato da:

$$B(\theta) = |\Phi(\theta)|. \tag{4.33}$$

In sintesi, i seguenti punti sono veri:

- ci sono K periodi occupati in $[0, T]$, con $\mathcal{B}_k = (\zeta_k, \eta_k(\theta))$, $k = 1, \cdots K$.
- $k \in \phi(\theta)$ se e solo se si verifica qualche *overflow* durante $\mathcal{B}_k$.

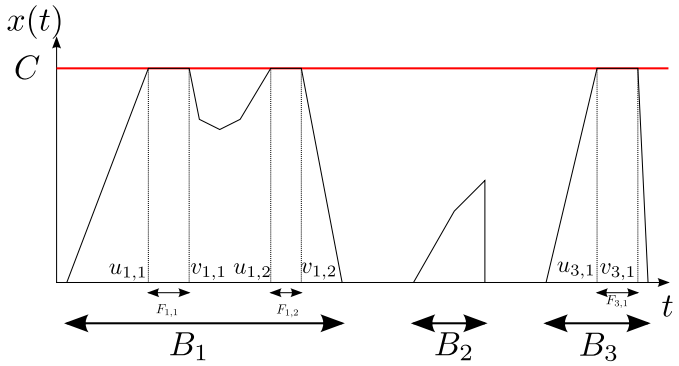


Figura 4.8: Esempio di evoluzione del buffer.

- Per ogni $k \in \phi(\theta)$ ci sono M_k periodi di *overflow* in $\mathcal{B}_k$.

Ad esempio in figura 4.8 sono presenti tre BPs ma solo in due di essi si verfica almeno un *overflow* quindi $\Phi(\theta) = \{1, 3\}$ e $B(\theta) = 2$. E' possibile dimostrare [5] che valgono le seguenti:

$$\frac{\partial L_T(\theta)}{\partial \theta} = -B(\theta) \tag{4.34}$$

e

$$\frac{\partial Q_T}{\partial \theta} = \sum_{k \in \phi(\theta)} (\eta_k(\theta) - u_{k,1}(\theta)). \tag{4.35}$$

Ossia il contributo di un periodo occupato a Q'_T è la lunghezza dell'intervallo definito dal primo punto nel quale il buffer diventa pieno e la fine del periodo occupato. Le 4.34 e 4.35 possono essere implementate dal seguente algoritmo.

Algoritmo 4.8 *[5]*

1. *si inizializzi un counter* $C := 0$ *e un timer cumulativo* $\mathcal{T} := 0$
2. *si inizializzi un timer* $\tau := 0$
3. *se si osserva un overflow all'istante* t *e* $\tau := 0$ *si ponga* $\tau := t$
4. *se un busy period termina all'istante* t *e* $\tau > 0$ *si ponga* $C := C - 1$ *e* $\mathcal{T} := \mathcal{T} + (t - \tau)$ *e* $\tau := 0$
5. *se* $t = T$ *e* $\tau > 0$ *si ponga* $C := C - 1$ *e* $\mathcal{T} := \mathcal{T} - \tau$

I valori finali di $\mathcal{T}$ e C forniscono L'_T e Q'_T

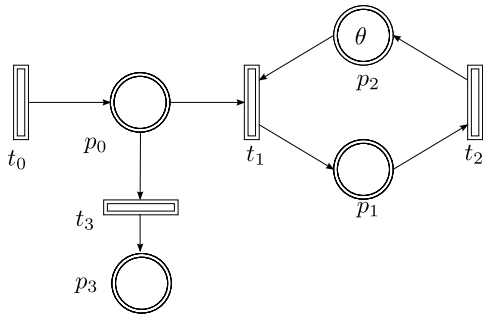


Figura 4.9: Modello del buffer con posto per tener conto dei pacchetti persi.

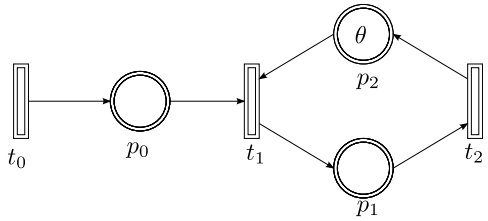


Figura 4.10: Modello semplificato del buffer.

4.7 Realizzazione modello tramite rete di Petri ibrida

Le reti di Petri ibride si prestano in maniera naturale alla modellizzazione di un buffer e grazie alla modifica apportata ad HYPENS sarà possibile simularne l'evoluzione in diversi contesti. Riferendoci alla figura 4.9

- la transizione t_0 modella l' *inflow rate*;
- la transizione t_1 va alla stessa velocità della transizione t_0 se il buffer non è pieno, altrimenti se è pieno va a velocità inferiore consentendo al posto p_1 di riempirsi;
- la transizione t_2 rappresenta l'*output flow* $v_2 = \delta(t)$;
- la transizione t_3 rappresenta il *loss* $v_3 = \gamma(t)$;
- la marcatura del posto p_0 è sempre nulla e modella un elemento connettore;
- la marcatura posto p_1: rappresenta lo spazio ancora disponibile nel buffer $m(p_1, t) = \theta - x(t)$;
- la marcatura del posto p_2: rappresenta il volume di fluido presente nel buffer $m(p_2, t) = x(t)$;

- la marcatura del posto p_3 rappresenta il numero totale di pacchetti persi all'istante τ,ossia $m(p_3, t) = \int_0^t \gamma(\theta, \tau) d\tau$.

In figura 4.9 è proposto un modello semplificato.

Capitolo 5

Esempi applicativi di IPA per reti di Petri ibride

In questo capitolo proponiamo alcuni esempi di simulazione di Reti di Petri ibride di particolare interesse nell'ambito dell'IPA.

5.1 Simulazione rete manifatturiera con tre macchine

In questa sezione si intende discutere i risultati ottenuti dalla simulazione della rete mostrata in figura 5.1 e verificarne la correttezza dell'evoluzione prima di applicarne la tecnica di ottimizzazione basata sull'IPA. Per verificare la correttezza della *function* si assumono tempi di guasto deterministici riportati nella tabella seguente:

	t_{guasto}s	$t_{rip.}$s
M_1	0.9	0.1
M_2	0.8	0.2
M_3	0.7	0.3

Si pone inoltre:

- $\theta_1 = \theta_2 = \theta_3 = \frac{1}{3}$;
- come tempo di simulazione $t_{stop} = 5s$;
- $c = [0\ 0\ 0\ 0\ 1]^T$.

Si può notare come lo stato discreto mostrato nelle figure 5.2(a), 5.2(b), 5.2(c) delle tre macchine evolva correttamente. Si noti come l'evoluzione delle velocità istantanee mostrate in figura 5.2(d), 5.3coincida con quanto descritto dalle equazioni 4.8 4.9, 4.10 4.11,4.12,4.13.

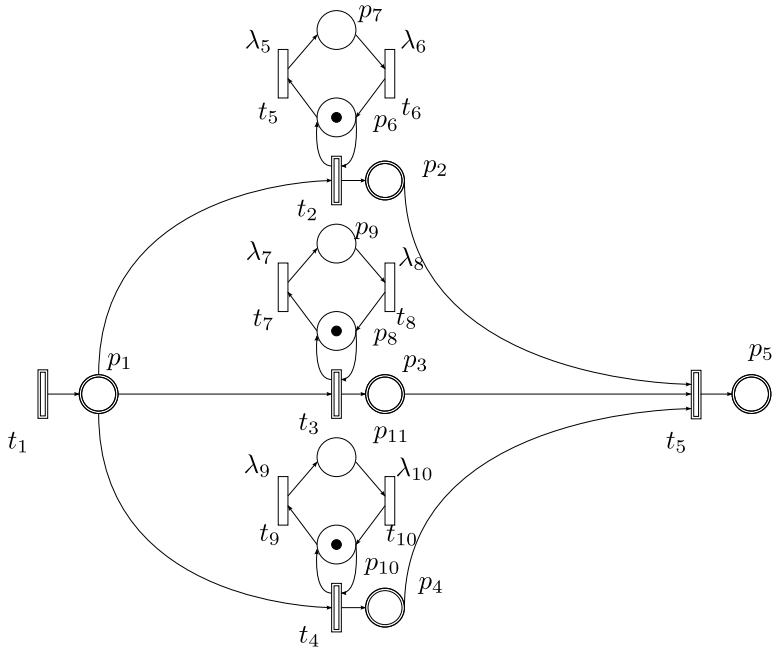
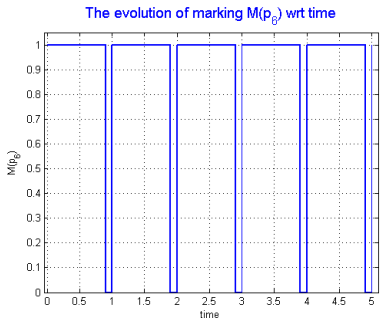


Figura 5.1: Rete considerata per la simulazione.

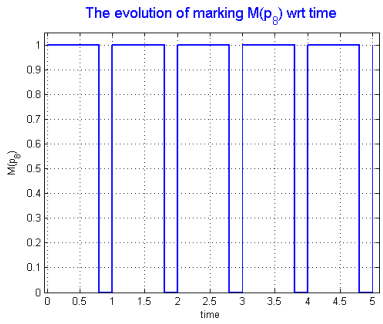
Osservazione 5.1 *Nello svolgere questa simulazione abbiamo utilizzato un approccio che si potrebbe definire* misto. *Abbiamo specificato in forma chiusa le velocità delle transizioni che dipendono dalla marcatura discreta della rete mentre abbiamo massimizzato la velocità della transizione* t_5 *(che dipende dalla marcatura continua). Questo, oltre a mostrare la flessibilità del simulatore modificato, pone un quesito interessante ossia se esista un vettore costo funzione della marcatura della rete* $\mathbf{c}(\mathbf{M})$ *che consenta di ottenere lo stesso risultato di simulazione avuto specificando parte delle velocità istantanee in forma chiusa. Gli esperimenti da noi condotti hanno dato tuttavia esito negativo.*

5.2 Simulazione buffer

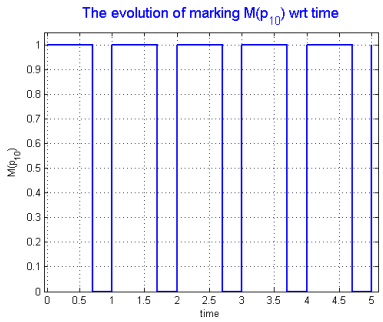
In questa sezione intendiamo verificare il corretto funzionamento del simulatore modificato analizzando l'evoluzione della marcatura e del vettore delle velocità istantanee della rete di Petri continua mostrata in figura 5.4 che modella un buffer con le relative perdite. I vincoli aggiuntivi per la scelta delle velocità istantanee delle transizioni continue sono specificati nel file *f_ab_m.m* della relativa *directory*. Tali vincoli sono scritti affinché vengano soddisfatte le equazioni del modello, ossia le 4.23, 4.24, 4.25. Per verificare la correttezza dell'evoluzione si propongono prima due esempi in cui sia l'*inflow rate* $\alpha(t)$, ovvero la velocità istantanea della transizione t_1, che il *service rate*



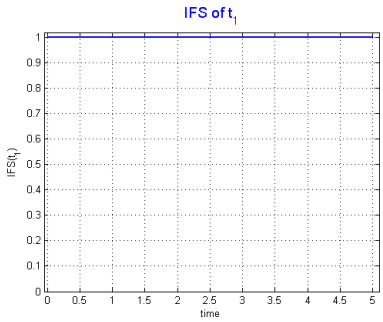
(a) Evoluzione marcatura discreta prima macchina.



(b) Evoluzione marcatura discreta seconda macchina.

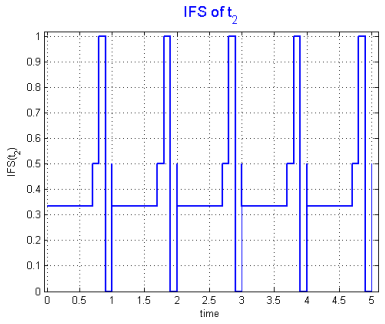


(c) Evoluzione marcatura discreta terza macchina.

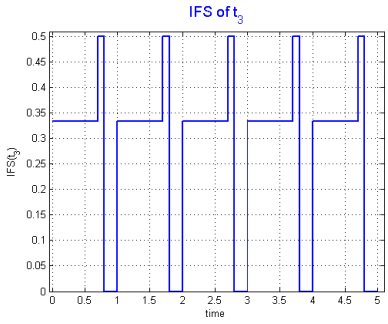


(d) Evoluzione velocità istantanea transizione di ingresso.

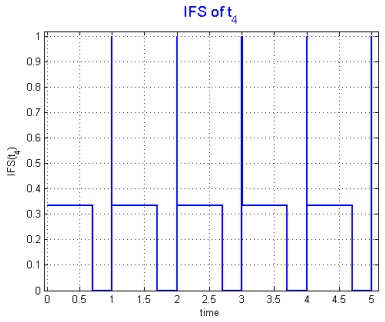
Figura 5.2



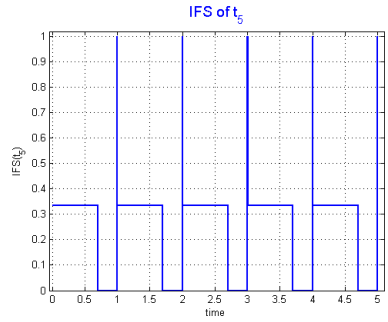
(a) Evoluzione velocità istantanea prima macchina.



(b) Evoluzione velocità istantanea seconda macchina.



(c) Evoluzione velocità istantanea terza macchina.



(d) Evoluzione velocità istantanea macchina assemblatrice.

Figura 5.3

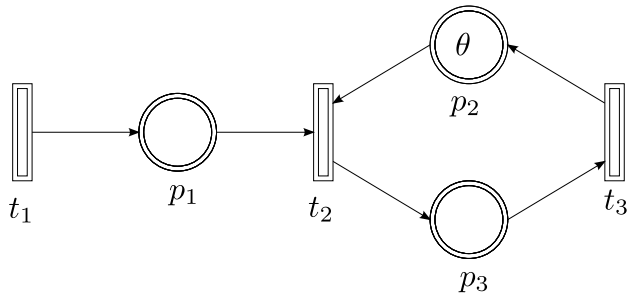


Figura 5.4: Rete di petri continua per modellare il buffer e le perdite.

$\beta(t)$, ovvero la velocità istantanea della transizione t_3 sono deterministici. Si noti che la transizione t_2 è una transizione che potremmo definire *fittizia* in quanto non ha un collegamento fisico immediato come le transizioni t_1 e t_3. La velocità della transizione t_2 è controllata in maniera tale che il posto p_1 si riempia al momento opportuno ossia quando si ha perdita di pacchetti. Nel primo esempio il *service rate* viene mantenuto costante mentre nel secondo viene fatto variare secondo una legge deterministica; come si vedrà, nel secondo esempio, si renderà necessaria una piccola modifica alla struttura della rete. Questi semplici esempi oltre a servire come verifica del corretto funzionamento del simulatore hanno anche lo scopo di mostrare come la modifica apportata consenta di specificare le velocità istantanee delle transizioni continue completamente in forma chiusa; la funzione $f\_ab\_m.m$ non fa altro che introdurre ulteriori vincoli nella forma di uguaglianza nell'insieme di ammissibilità del problema di programmazione lineare. Per ultimo, nel terzo esempio, simuleremo il sistema con una sorgente più realistica. Tutto il codice è riportato in appendice.

5.2.1 Simulazione buffer: *inflow rate* e *outflow rate* costanti

Come detto precedentemente, in questo primo esempio assumiamo che sia l'*inflow rate* che il *service rate* siano costanti per tutta la durata della simulazione; con questo primo semplice esempio sarà possibile verificare facilmente il corretto funzionamento del simulatore modificato. I vincoli imposti saranno dunque i seguenti:

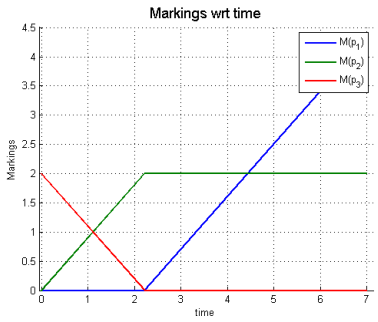
- *inflow rate* $v_1(t) = \alpha(t) = 1$;
- *service rate* $v_3(t) = \beta(t) = \frac{1}{10}$;
- capacità del buffer $M_{p_3}(0) = \theta = 2$;
- durata della simulazione $time\_stop = 2$ s.

5.2.1.a Analisi dei risultati della simulazione

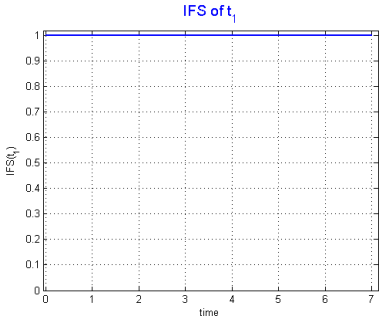
Ovviamente essendo l' *inflow rate* sempre maggiore del *service rate*, ci aspettiamo che il *buffer*, ossia la marcatura del posto p_2 si riempia ad una velocità costante pari a $v_2(t) - v_1(t) = 0.9$ s^{-1} per un intervallo di tempo pari a $\frac{\theta}{v_2(t)-v_1(t)} = 2.22$ s e che, in maniera duale, comincino ad aversi perdite, e quindi cominci a riempirsi il posto p_1 solo dopo un intervallo di tempo pari a 2.22 s. Formalmente:

$$\frac{dM_{p_2}}{dt} = \begin{cases} 0.9 & \text{se } t \leq 2.22 \\ 0 & \text{altrimenti} \end{cases} \tag{5.1}$$

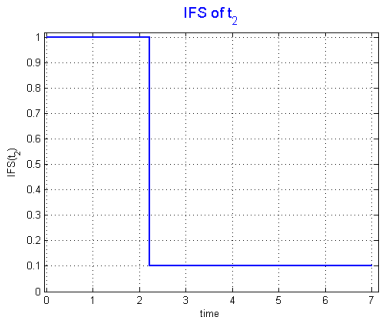
$$\frac{dM_{p_1}}{dt} = \begin{cases} 0 & \text{se } t \leq 2.22 \\ 0.9 & \text{altrimenti.} \end{cases} \tag{5.2}$$



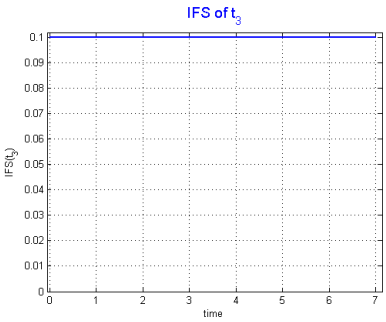
(a) Evoluzione marcatura continua.



(b) Evoluzione velocità istantanea transizione t_1.



(c) Evoluzione velocità istantanea transizione t_2.



(d) Evoluzione velocità istantanea transizione t_2.

Figura 5.5

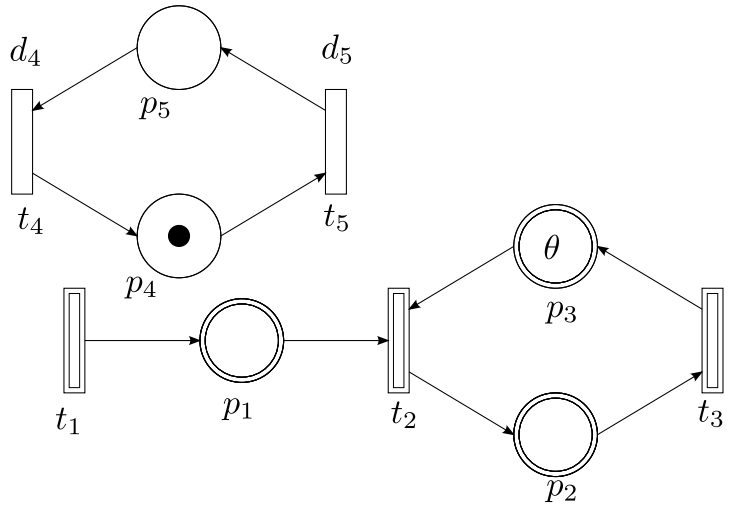


Figura 5.6: Rete utilizzata per la simulazione nel caso di *outflow* rate variabile.

Dall'analisi della figura 5.5 se ne deduce che le condizioni 5.1, 5.2 risultano essere soddisfatte; ovviamente anche l'evoluzione del vettore delle velocità istantanee mostrato in figura risulta coerente con quanto imposto. Infatti, v_1 e v_3 seguono l'andamento imposto, mentre per v_2 si ha il seguente andamento:

$$v_2 = \begin{cases} 1 & \text{se } t \leq 2.22 \text{ s} \\ 0.1 & \text{altrimenti} \end{cases} \tag{5.3}$$

risultando coerente con l'evoluzione della marcatura del posto p_1.

5.2.2 Simulazione buffer: *inflow rate* costante, *service rate* periodico

Ora si considera un caso leggermente più complicato in cui l'*inflow rate* viene mantenuto costante mentre il *service rate* viene fatto variare periodicamente.

Grazie alle modifiche apportate al software sarà possibile simulare tale comportamento, a patto di introdurre, nel modello della rete, delle transizioni discrete temporizzate e dei posti discreti la cui marcatura consenta di generare correttamente i vincoli per la scelta del vettore di velocità istantanee corretto. La rete che sarà elaborata dal simulatore è quella mostrata in figura 5.6. I vincoli imposti sono i seguenti:

- *inflow rate* $\alpha(t) = 1$
- *service rate*

$$\beta(t) = \begin{cases} 0.1 & se \quad (n-1)\cdot 3 \leq t \leq n \cdot 3 \\ 1.1 & se \quad n \cdot 3 < t \leq (n+1)\cdot 3 \end{cases} \quad n = 1, 2 \cdots \tag{5.4}$$

- capacità del buffer $\theta = 2$
- durata della simulazione $time\_stop =$ 30 s.

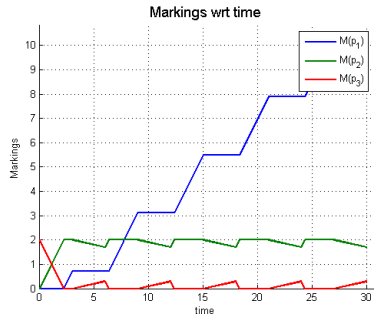
5.2.2.a Analisi dei risultati della simulazione

In questo caso, in alcuni intervalli temporali l'*inflow rate* è maggiore del *service rate* e quindi ci aspettiamo che il buffer si riempia oppure si comincino a perdere pacchetti se esso è già pieno. Negli intervalli temporali invece dove l'*inflow rate* è minore dell'*service rate* ci aspettiamo o che il buffer progressivamente si svuoti oppure, se è già vuoto, che l'*outflow rate* diventi uguale all'*inflow rate*. Inizialmente il *service rate* è inferiore all' *inflow rate* quindi ci aspettiamo che progressivamente il *buffer* si riempia il tempo impiegato affinchè il buffer si riempia sino alla sua capacità massima è pari, come nel caso precedente, a 2.22 s quando questo avviene l'*inflow rate* è ancora maggiore del *service rate* e quindi cominceranno ad aversi perdite; all'istante di tempo $t = 3$ s il *service rate* diventa maggiore dell' *inflow rate* e quindi il *buffer* comincerà a svuotarsi raggiungendo un occupazione pari a $M_{p2}(6) = 1.7$ a questo punto il *service rate* torna ad essere inferiore dell'*inflow rate* il tempo necessario affinché il *buffer* si riempia totalmente è pari a $\Delta_t = \frac{M_{p3}(0) - M_{p2}(6)}{v_1(t) - v_3(t)} = 0.33$ s ovviamente questo risultato si sarebbe dovuto scartare se si fosse ottenuto un intervallo temporale superiore a quello in cui la velocità di scatto della transizione t_3 rimane costante; poiché in questo caso non avviene, possiamo affermare che all'istante temporale $t = 6.33$ s il *buffer* torna ad essere pieno e sino a quando risulta $v_1(t) > v_3(t)$, ovvero sino all'istante di tempo $t = 9$ s, continueranno ad accumularsi pacchetti persi ad una velocità pari a $v_1(t) - v_2(t) = 0.9$ s^{-1}. Le figure 5.7 mostrano come il sistema evolva correttamente in base all'analisi precedentemente descritta.

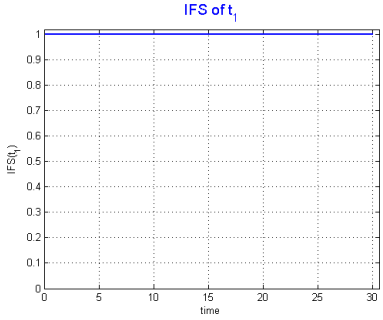
5.2.3 Simulazione buffer: *inflow rate burst*, *outflow rate* costante

In questo terzo esempio vogliamo simulare il sistema con una sorgente di tipo ON-OFF. Durante il periodo ON la velocità della transizione t_1 assume valore unitario mentre, durante il periodo OFF, la velocità della transizione t_1 è nulla poiché non arriva alcun pacchetto.

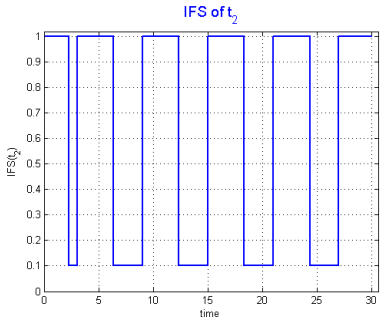
Questo esempio vuole mostrare come il modello proposto per il *buffer* possa essere utilizzato anche per simulare situazioni in cui o il traffico della rete o i tassi di servizio non siano noti in maniera deterministica ma li si conosca solo su base statistica. Ovviamente è molto importante che si possa poter simulare in un contesto del genere perché è il caso che si verifica più spesso nella pratica. Il modello utilizzato nella simulazione è quello mostrato in figura 5.8. Si noti che l'unica differenza con la rete utilizzata per la simulazione dell'esempio precedente è che la temporizzazione delle transizioni



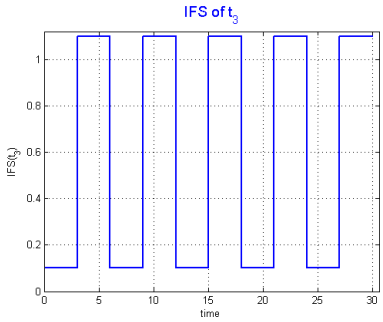
(a) Evoluzione marcatura continua.



(b) Evoluzione velocità istantanea transizione t_1.



(c) Evoluzione velocità istantanea transizione t_2.



(d) Evoluzione velocità istantanea transizione t_2.

Figura 5.7

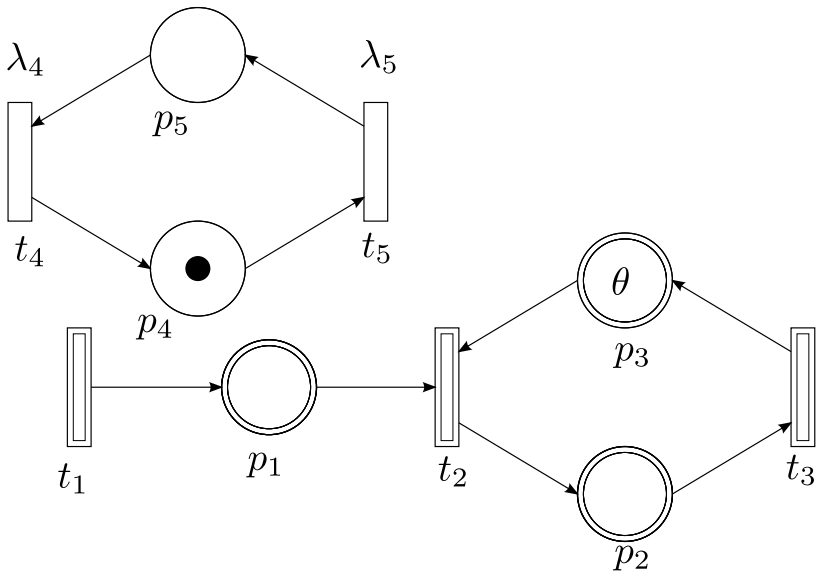


Figura 5.8: Rete di Petri ibrida (terzo esempio).

t_4 e t_5 non è più deterministica ma esponenzialmente distribuita. Quando il posto p_4 è marcato vuol dire che ci troviamo in un periodo ON; viceversa, quando il posto p_4 è vuoto vuol dire che ci troviamo in un periodo OFF. In base al valore della marcatura discreta assunto dalla rete, il simulatore genererà il vincolo opportuno affinché la velocità della transizione t_1 sia corretta.

Ipotizziamo che il traffico sia di tipo intermittente (*bursty*). E' un tipo di traffico che si ha quando il flusso di dati è caratterizzato da brevi periodi di trasmissione intervallati da lunghi intervalli di inattività. Il traffico di dati nelle *Local Area Networks (LAN)* è tipicamente di tipo intermittente. Un altro esempio può essere lo scambio di dati in un accesso interattivo ad un database remoto; infatti, in questo caso, la trasmissione utile è molto discontinua nel tempo. Dal punto di vista delle reti il flusso intermittente prevede un tempo medio di utilizzo della risorsa limitato. I bit consecutivi trasmessi dalle sorgenti sono detti *burst*. Il rapporto fra la somma delle durate dei bit emessi (raffiche consecutive) e la durata totale della trasmissione costituisce il tasso d'attività della sorgente.

In base a queste considerazioni possiamo imporre, per la nostra simulazione, che il periodo ON sia mediamente in rapporto di $\frac{1}{10}$ con il periodo ON-OFF medio totale. Poniamo quindi:

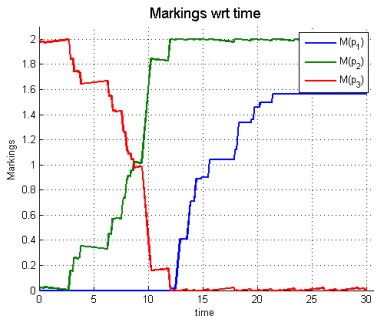
- $\lambda_4 = 0.9$;
- $\lambda_5 = 0.1$;
- *inflow rate*:
$$v_1(t) = \begin{cases} 1 & \text{se in periodo ON} \\ 0 & \text{altrimenti;} \end{cases} \tag{5.5}$$
- *service rate* $v_3(t) = 0.01$;
- dimensione del buffer $\theta = 2$;
- durata della simulazione time_stop = 30 s.

5.2.3.a Analisi dei risultati delle simulazioni

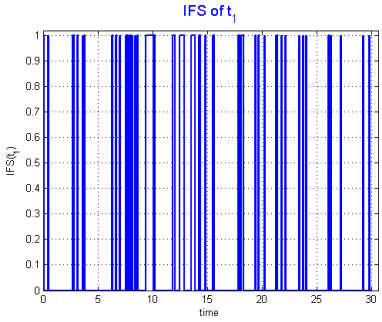
I risultati di questa terza simulazione relativa all'evoluzione del *buffer* sono mostrati nelle figure 5.9. Osservando la figura 5.9 è possibile notare come i periodi ON durino molto meno dei periodi OFF. Questo è quello che ci attendevamo.

5.2.4 Stima gradiente della perdita e della lunghezza media della coda

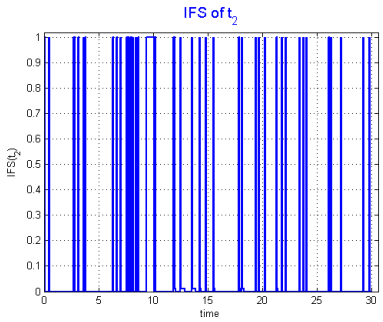
E' stato implementato in *MATLAB* l'algoritmo 4.8 per la stima del gradiente delle funzioni di prestazione. Il codice è riportato in appendice. Si



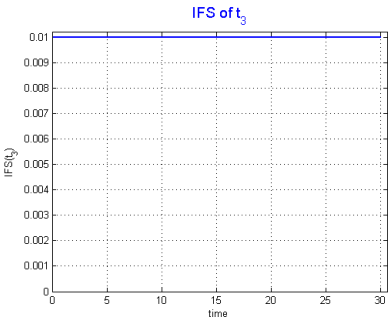
(a) Evoluzione marcatura continua.



(b) Evoluzione velocità istantanea transizione t_1.



(c) Evoluzione velocità istantanea transizione t_2.



(d) Evoluzione velocità istantanea transizione t_2.

Figura 5.9

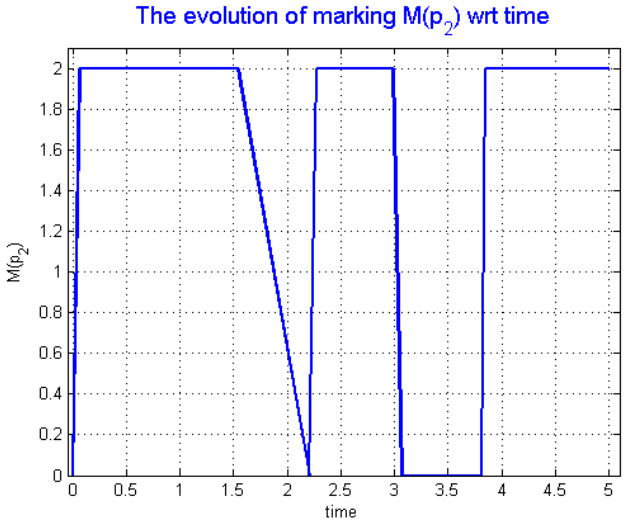


Figura 5.10: Evoluzione del contenuto del buffer.

vuole, tramite un esempio, mostrarne il corretto funzionamento. Si consideri nuovamente la rete in figura 5.8. Si è posto:

- $\lambda_4 = \lambda_5 = 0.5$;
- *inflow rate* scelto casualmente ad ogni macro-evento (interi compresi tra 1 e 100);
- *outflow rate* $\beta(t) = 30$ per tutta la durata della simulazione;
- dimensione del buffer $\theta = 2$;
- durata della simulazione time_stop = 5 s.

5.2.4.a Analisi dei risultati delle simulazioni

Le stime fornite dall'algoritmo sono le seguenti:

$$L'_T = -3;$$

e

$$Q'_T = 4.0916.$$

Analizzando la figura 5.10 e si può notare che:

1. ci sono tre periodi occupati nei quali avviene almeno un *overflow*;
2. sommando per ogni periodo occupato gli intervalli di tempo che vanno dall'inizio di un *overflow* alla fine del periodo occupato si ottiene circa 4 s .

Possiamo dunque affermare che l'algoritmo funziona correttamente e fornisce stime esatte.

5.3 Ottimizzazione parametri di *routing*

Grazie alla modifica del *tool* è possibile utilizzare *HYPENS* per l'ottimizzazione di parametri di *routing* utilizzando l'algoritmo 4.7. Si sono effettuate due simulazioni. Nella prima i tassi di guasto e riparazione delle macchine sono stati supposti identici ($\lambda_{1,f} = \lambda_{2,f} = \lambda_{3,f} = 0.9$ e $\lambda_{1,r} = \lambda_{2,r} = \lambda_{3,r} = 0.1$) mentre nel secondo caso una macchina sta per molto più tempo guasta rispetto alle altre ($\lambda_{1,f} = \lambda_{2,f} = 0.9$,$\lambda_{1,r} = \lambda_{2,r} = 0.1$ e $\lambda_{3,f} = 0.1$ e $\lambda_{3,r} = 0.9$). I

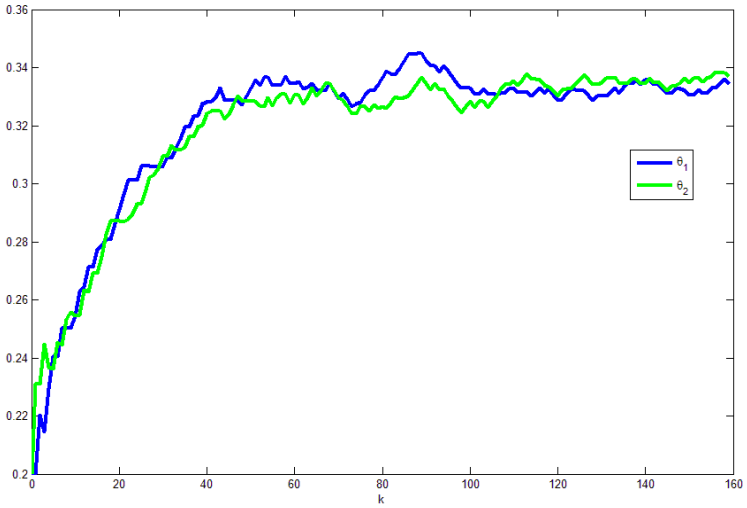


Figura 5.11: Valori di $\theta_1(k)$ e $\theta_2(k)$ nel primo caso (le tre macchine hanno gli stessi tassi di guasto e riparazione).

risultati delle simulazioni sono mostrati nelle figure 5.11 e 5.12 sono coerenti con quanto ottenuto nell'articolo [2]. L'algoritmo utilizzato per la stima dei gradienti è stato implementato da C. Seatzu [2] ed è stato adattato per poter essere utilizzato con i dati forniti da *HYPENS*.

5.4 Pesi della funzione obbiettivo variabili

Si ricorda che ad ogni macro-evento occorre risolvere un problema di programmazione lineare in cui la funzione obbiettivo da massimizzare è:

$$J = \mathbf{c}^T \mathbf{v} \tag{5.6}$$

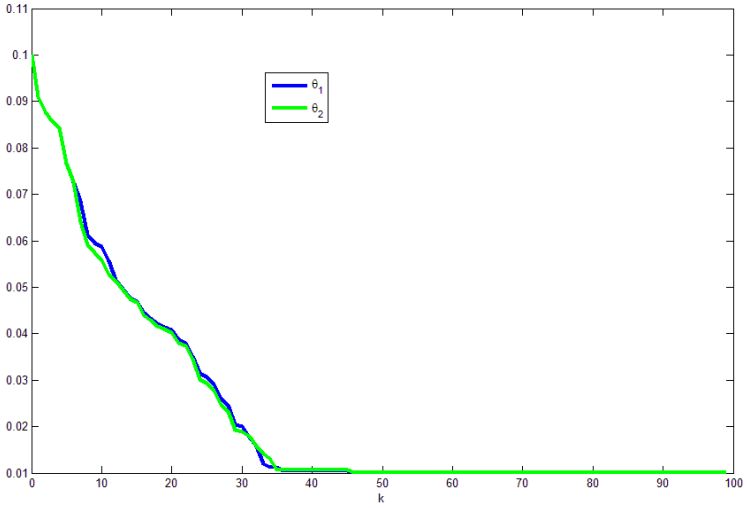


Figura 5.12: Valori di $\theta_1(k)$ e $\theta_2(k)$ nel secondo caso (la terza macchina ha tasso di guasto superiore rispetto alle altre due).

dove le componenti del vettore **c** rappresentano i pesi associati alle transizioni, quindi, hanno il compito di assegnare un livello di priorità. Si consideri nuovamente la rete in figura 5.1 dove questa volta assumiamo il posto p_1 vuoto e che la terza macchina sia per molto più tempo guasta rispetto alle altre due. Le simulazioni hanno portato i risultati nella seguente tabella (tempo di simulazione 15 secondi e tempi di guasto deterministici). Dove:

c	$\overline{v_5}$
$[00001]^T$	0.0313
$[00011]^T$	0.0988
$[00111]^T$	0.0459
$[01111]^T$	0.0459
$[11111]^T$	0.0612
$[11121]^T$	0.0857
$[11122]^T$	0.0395

Tabella 5.1: Velocità di scatto media della transizione di uscita al variare del vettore **c** con tempi di guasto deterministici.

$$\overline{v_5} = \frac{1}{T}\int_0^T v_5(t)dt \tag{5.7}$$

è il valore medio della velocità di scatto della transizione t_5. Analizzando il risultato delle simulazioni se ne deduce facilmente che la scelta del vettore **c**

influenzi la velocità di scatto media della transizione di uscita. Si può quindi pensare di far variare il vettore $\mathbf{c}$ in funzione della marcatura raggiunta dalla rete secondo una legge arbitraria.In questo caso particolare il valore di $\overline{v_5}$ ottenuto con la scelta del vettore $\mathbf{c} = [00011]^T$ coincide con quello che si otterrebbe imponendo i vincoli di *routing* e ponendo $\theta_1 = \theta_2 = 0$. In questo caso quindi, l'ottimizzazione globale della funzione obbiettivo $J = \int_0^T v_5(t)dt$ coincide con l'ottimizzazione miopica della funzione obbiettivo $J = v_4 + v_5$.

Capitolo 6

Conclusioni

In questa tesi ci siamo proposti di indagare su alcune tecniche di analisi perturbativa infinitesimale applicata alle reti di Petri ibride. Poiché l'analisi perturbativa infinitesimale è una tecnica semi-analitica era necessario:

1. disporre di un software che consentisse di simulare una qualsiasi rete di Petri ibrida;
2. che questa evoluzione potesse essere vincolata a dei parametri da ottimizzare;
3. implementare in linguaggio di programmazione alcuni algoritmi per la stima del gradiente delle funzioni di prestazione.

Per quanto riguarda i primi due punti, abbiamo deciso di utilizzare un *tool* già esistente (*HYPENS*) ed ampliarne in maniera opportuna le potenzialità per i nostri scopi. Abbiamo fatto in modo quindi che l'evoluzione della rete potesse essere specificata in forma chiusa e vincolata a dei parametri. Abbiamo studiato alcune reti di particolare interesse applicativo nell'ambito dell'analisi perturbativa infinitesimale:

1. una rete rappresentante un sistema manifatturiero con tre macchine;
2. una rete che modella il processo di arrivo di pacchetti in un *buffer* e le relative perdite;

Per quanto riguarda la prima rete:

1. abbiamo specificato in MATLAB i vincoli inerenti l'evoluzione;
2. abbiamo mostrato come il modello evolva correttamente;
3. abbiamo utilizzato l'algoritmo di Robbins-Monro implementato da C.Seatzu per simulare alcuni casi di ottimizzazione dei parametri e se ne è verificata la correttezza;

4. abbiamo generalizzato la rete e le formule necessarie alla stima del gradiente della funzione di prestazione al caso di n macchine;

5. abbiamo confrontato la tecnica dell'ottimizzazione tramite IPA con una tecnica di natura euristica.

Per quanto riguarda la seconda rete:

1. abbiamo specificato in MATLAB i vincoli inerenti l'evoluzione;

2. abbiamo mostrato come il modello evolva correttamente anche con svariati tipi di sorgente;

3. abbiamo implementato l'algoritmo per la stima del gradiente della funzione di prestazione e verificata la correttezza.

   Fra i possibili sviluppi futuri, vi è sicuramente lo studio di formule di analisi perturbativa infinitesimale per sistemi modellizzabili tramite le reti di Petri ibride e a cui possano essere applicate formule di ottimizzazione.

Appendice A

Codice

In questa appendice si riporta il codice che è stato sviluppato o modificato.

A.1 simulator1_HPN

```
function [Type,M_Evol,IFS_Evol,P_macro,Event_macro_Evol,firing_transition,
timer_macro_event,tau,Q_Evol,timer,nc_nd_qc_qd]=...
    simulator1_HPN
(Pre,Post,M,vel,v,D,s,alpha,time_stop,simulation_type,it,c_m,ab_m,theta,c)
%
% simulator1_HPN : This function simulates a Hybrid Petri Net
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Codice non modificato
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %modifiche
        if(ab_m)
        [a1 b1 theta const_add]=f_ab_m(M,theta);
         a=cat(1,a,a1);
         b=cat(1,b,b1);
         [rb1 cb1]=size(b1);
         for i=1:rb1
         ctype=cat(1,ctype,const_add);
         end
        end
        if(c_m)
           [c]=f_c_m(M);
        end
        if(cont_places_const~=-1)
         [r size_cont_places_const]= size(cont_places_const);
         for i=1:size_cont_places_const
             ctype=cat(1,ctype,'S');
```

```
            b=cat(1,b,0);
            a=cat(1,a,constrains_p(cont_places_const(i),:));
         end

        [IFS M2]=glpkmex(set,c,a,b,ctype,lb,ub,vartype);
        a=a(1:(end-(rb1+size_cont_places_const)),:);
        b=b(1:(end-(rb1+size_cont_places_const)));


    else
        IFS=NaN;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Codice non modificato
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.2 Funzione per il controllo del buffer con beta costante

```
%Funzione che genera gli IFS per una coda
function [a1 b1 theta const_add]=f_ab_m(M_in,theta)
const_add=['S' ]';%specifica il tipo di vincolo 'S' è = (si veda help glpkmex)
%Questa funzione genera una sequenza di vettori IFS a seconda della
%marcatura raggiunta dalla rete.
[r ,c]=size(M_in);
%Specificare il numero di posti continui e il numero di posti discreti
pc=3;
pd=2;
%epsilon=1;
%Seleziono la parte discreta della marcatura
M_in_D=M_in(pc+1:pc+pd);
%Seleziono la parte continua della marcatura
M_in_C=M_in(1:pc);
alfa=1;
beta=0.1;
%Specifico gli IFS generabili
%A seconda della marcatura della rete scelgo l'IFS
if(M_in_C(2) > 10*eps)
a1=eye(3);
b1=[alfa 0 beta]';
else
```

```
if(alfa <=beta)
a1=eye(3);
b1=[alfa 0 alfa]';
else
a1=eye(3);
b1=[alfa 0 beta]';
end
end
if ( M_in_C(2) >= (theta-10*eps))
if(alfa > beta)
b1=b1+[0 beta 0]';
else
b1=b1+[0 alfa 0]';
end
else
b1=b1+[0 alfa 0]';
end
end
```

A.3 Funzione per il controllo del buffer con beta variabile

```
%Funzione che genera gli IFS per una coda
function [a1 b1 theta const_add]=f_ab_m(M_in,theta)
const_add=['S' ]';%specifica il tipo di vincolo 'S' è = (si veda help glpkmex)
%Questa funzione genera una sequenza di vettori IFS a seconda della
%marcatura raggiunta dalla rete.
[r ,c]=size(M_in);
%Specificare il numero di posti continui e il numero di posti discreti
pc=3;
pd=2;
%Seleziono la parte discreta della marcatura
M_in_D=M_in(pc+1:pc+pd);
%Seleziono la parte continua della marcatura
M_in_C=M_in(1:pc);
alfa=1;
beta1=1.1;
beta2=0.1;
%Specifico gli IFS generabili
%A seconda della marcatura della rete scelgo l'IFS
if(M_in_D(1)==1)
beta=beta1;
else
```

```
beta=beta2;
end
if(M_in_C(2) > 10*eps)
a1=eye(3);
b1=[alfa 0 beta]';
else
if(alfa <=beta)
a1=eye(3);
b1=[alfa 0 alfa]';
else
a1=eye(3);
b1=[alfa 0 beta]';
end
end
if ( M_in_C(2) >= (theta-10*eps))
if(alfa > beta)
b1=b1+[0 beta 0]';
else
b1=b1+[0 alfa 0]';
end
else
b1=b1+[0 alfa 0]';
end
end
```

A.4 Funzione per il controllo del buffer con traffico *burst*

```
function [a1 b1 theta const_add]=f_ab_m(M_in,theta)
const_add=['S' ]';%specifica il tipo di vincolo 'S' è = (si veda help glpkmex)
%Questa funzione genera una sequenza di vettori IFS a seconda della
%marcatura raggiunta dalla rete.
[r ,c]=size(M_in);
%Specificare il numero di posti continui e il numero di posti discreti
pc=3;
pd=2;
%Seleziono la parte discreta della marcatura
M_in_D=M_in(pc+1:pc+pd);
%Seleziono la parte continua della marcatura
M_in_C=M_in(1:pc);
beta=0.01;
if(M_in_D(1)==1)
    alfa=1;
```

```
else
    alfa=0;
end
if(M_in_C(2) > 10*eps) %se il buffer non è vuoto
    a1=eye(3);
    b1=[alfa 0 beta]';
else
    if(alfa <=beta)
        a1=eye(3);
        b1=[alfa 0 alfa]';
    else
         a1=eye(3);
        b1=[alfa 0 beta]';

    end
end
if ( M_in_C(2) >= (theta-10*eps)) %se il buffer è pieno
    if(alfa > beta)
        b1=b1+[0 beta 0]';
    else
        b1=b1+[0 alfa 0]';
    end
else
    b1=b1+[0 alfa 0]';
end
end
```

A.5 Funzione per il controllo del buffer con traffico casuale

```
function [a1 b1 theta const_add]=f_ab_m(M_in,theta)
const_add=['S' ]';%specifica il tipo di vincolo 'S' è = (si veda help glpkmex)
%Questa funzione genera una sequenza di vettori IFS a seconda della
%marcatura raggiunta dalla rete.
[r ,c]=size(M_in);
%Specificare il numero di posti continui e il numero di posti discreti
pc=3;
pd=2;
%Seleziono la parte discreta della marcatura
M_in_D=M_in(pc+1:pc+pd);
%Seleziono la parte continua della marcatura
M_in_C=M_in(1:pc);
beta=30;
```

```
if(M_in_D(1)==1)
    alfa=randi(100,1,1);
else
    alfa=randi(100,1,1);
end
if(M_in_C(2) > 10*eps) %se il buffer non è vuoto
    a1=eye(3);
    b1=[alfa 0 beta]';
else
    if(alfa <=beta)
        a1=eye(3);
        b1=[alfa 0 alfa]';
    else
         a1=eye(3);
        b1=[alfa 0 beta]';

    end
end
if ( M_in_C(2) >= (theta-10*eps)) %se il buffer è pieno
    if(alfa > beta)
        b1=b1+[0 beta 0]';
    else
        b1=b1+[0 alfa 0]';
    end
else
    b1=b1+[0 alfa 0]';
end
end
```

A.6 IPA_buffer(x,theta,dt)

```
function [DLT, DQT]=IPA_buffer(x,theta,t)
%Stima il gradiente della perdita e della lunghezza media della coda
%rispetto alle dimensioni del buffer .
%x-vettore contenente l'evoluzione della quantita' di pacchetti presenti
%nel buffer.
%theta-dimensione del buffer.
%t-vettore contenente gli istanti di tempo in cui si verificano i
%macro-eventi.
%DLT- Derivata Loss Volume
(numero di busy periods in cui si verifica almeno un overflow)
%DQT- Derivata Cumulative Workloadn=size(x);
n=n(2);
```

```
C=0; %inizializzo il counter
T=0; %timer cumulativo
tau=0;%timer
epsilon=0.01;

for i=1 :n
    if((x(i) >= theta-epsilon)& (tau==0))
        tau=t(i);
     end
    if((x(i)<=epsilon) & (tau>0))
        C=C-1;
        T=T+(t(i)-tau);
        tau=0;
    end
end
if(tau >0)
    C=C-1;
    T=T+(t(i)-tau);
end
DLT=C;
DQT=T;
end
```

A.7 Funzione per la generazione dei vincoli nella simulazione della rete manifatturiera

```
function [a1 b1 theta const_add]=f_ab_m(M,theta);
%Evoluzione rete manifatturiera
const_add=['S' ]';%specifica il tipo di vincolo 'S' è = (si veda help glpkmex)
theta1=theta(1,1);
theta2=theta(2,1);
theta3=1-theta1-theta2;
MD=M(6:end);
a1=[];
b1=[];
if ((MD(1)==1)&(MD(5)==1)&(MD(3)==0)) %v1 ,v3 on v2 off
a1=[zeros(1,5); [zeros(3,1) eye(3) zeros(3,1)] ;zeros(1,5)];
b1=[0 (theta1+(theta2*theta1)/(theta1+theta3))...
0 (theta3+(theta2*theta3)/(theta1+theta3)) 0]';
else
if ((MD(1)==1)&(MD(3)==1)&(MD(5)==0)) %v1 v2 on,v3 off
a1=[zeros(1,5); [zeros(3,1) eye(3) zeros(3,1)] ;zeros(1,5)];
b1=[0 (theta1+(theta3*theta1)/(theta1+theta2))...
```

```
 (theta2+(theta3*theta2)/(theta1+theta2)) 0 0]';
else
if((MD(3)==1)&(MD(5)==1)&(MD(1)==0))%v2 v3 on ,v1 off
a1=[zeros(1,5); [zeros(3,1) eye(3) zeros(3,1)] ;zeros(1,5)];
b1=[0 0 (theta2+(theta1*theta2)/(theta2+theta3))...
 (theta3+(theta1*theta3)/(theta2+theta3)) 0]';
end
end
end
if ((MD(1)==1)&(MD(3)==1)&(MD(5)==1)) %v1 v2 v3 on
a1=[zeros(1,5); [zeros(3,1) eye(3) zeros(3,1)] ;zeros(1,5)];
b1=[0 theta1 theta2 theta3 0]';
end
if ((MD(1)==0)&(MD(3)==0)&(MD(5)==0))
a1=[zeros(1,5); [zeros(3,1) eye(3) zeros(3,1)] ;zeros(1,5)];
b1=[0 0 0 0 0]';
end
```

A.8 Funzione per la simulazione della rete di imbottigliamento

```
%Simuazione impianto di imbottigliamento.

%Matrici relative al modello della rete.
Pre_CC=[0 0 1;0 0 0];
Post_CC=[1 1 0;0 0 1];
Pre_CD=[0 0 0 0 0 0;0 0 0 0 0 0];
Post_CD=[0 0 0 0 0 0;0 0 0 0 0 0];
Pre_DC=[0 1 0; 0 0 0; 1 0 0;0 0 0;0 0 1;0 0 0];
Post_DC=[0 1 0;0 0 0; 1 0 0;0 0 0;0 0 1;0 0 0];
Pre_DD=eye(6);
Post_DD=[0 1 0 0 0 0;1 0 0 0 0 0;0 0 0 1 0 0;
0 0 1 0 0 0;0 0 0 0 0 1;0 0 0 0 1 0];
%Marcatura Iniziale della rete.
M0_C=[0 0];
M0_D=[1 0 1 0 1 0];
%Matrice contenente in ogni riga velocità minima e massima di ciascuna
%transizione continua
vel=[2 5;
     4 10;
     3 11];
 %Vettore che specifica la temporizzazione (deterministica o stocastica)
 %per ciascuna transizione discreta.
```

```
 v=[1 1 1 1 1 1];
 %Matrice che quantifica la temporizzazione
 D=[0.1 NaN NaN;
    0.9 NaN NaN;
    0.2 NaN NaN;
    0.9 NaN NaN;
    0.3 NaN NaN;
    0.9 NaN NaN];
%Numero di serventi.
s=[0 0 0 0 0 0];
%Priorità.
alpha=[1 1 1 1 1 1];
it=1;
[Pre,Post,M0,vel,v,D,s,alpha]=make_HPN...
(Pre_CC,Pre_CD,Pre_DC,Pre_DD,Post_CC,Post_CD,Post_DC,Post_DD,...
    M0_C,M0_D,vel,v,D,s,alpha,it);
time_stop=30;
simulation_type=0;
it=1;
c_m=0;
ab_m=0;
theta=0;
%Vettore in cui vanno specificati i posti la cui marcatura deve rimanere
%costante
cont_places_const=[1 ];
c=[1 1 1];
[Type,M_Evol,IFS_Evol,P_macro,Event_macro_Evol...
,firing_transition,timer_macro_event,tau,Q_Evol,...
    timer,nc_nd_qc_qd]=...
    simulator1_HPN...
    (Pre,Post,M0,vel,v,D,s,alpha,time_stop,...
    simulation_type,it,c_m,ab_m,theta,cont_places_const,c);

statistic_plot=1;
graph=1;
marking_plot=[1 2 ];
Td_firing_plot=1;
up_marking_plot=[1 2 ];
Pd_prob_plot=0;
Pd_ave_t_plot=[-1];
IFS_plot=[1 2 3];
up_IFS_plot=[-1];
IFS_ave_plot=0;
Td_freq_plot=0;
```

```
Md_freq_en=1;
 Evol={Type,M_Evol,IFS_Evol,P_macro,Event_macro_Evol,...
 firing_transition,timer_macro_event,tau,Q_Evol,nc_nd_qc_qd};
[P_ave, P_max, IFS_ave, Td_ave, Pd_freq, Md_ave_t, Md_freq]=...
    analysis_HPN(Evol, statistic_plot, graph, marking_plot, Td_firing_plot,...
    up_marking_plot, Pd_prob_plot, Pd_ave_t_plot, IFS_plot,...
     up_IFS_plot, IFS_ave_plot,Td_freq_plot,Md_freq_en,it);
```

A.9 Funzione per l'ottimizzazione dei parametri di routing

```
function [theta1,theta2,dJ1,dJ2] = ...
opt_theta1(theta1_0,theta2_0,C,max_step);
theta1(1)=theta1_0;
theta2(1)=theta2_0;
theta=[theta1(1) theta2(1)]';
%implements the gradient algorithm
rete_con_posti_fittizi
%marcatura iniziale dei posti continui
M0_C=[50 0 0 0 0];
%
M0_D=[1 0 1 0 1 0 1 0];
vel=[0 1;0 1;0 1;0 1;0 1];
v=[2 2 2 2 2 2 1 1];
D=[0.9 NaN NaN;
   0.1 NaN NaN;
   0.9 NaN NaN;
   0.1 NaN NaN;
   0.1 NaN NaN;
   0.9 NaN NaN;
   0.01 NaN NaN;
   0.01 NaN NaN;
  ];
s=[0 0 0 0 0 0 0 0 ];
alpha=[1 1 1 1 1 1 1 1];
it=1;
%J=[0 0 0 0 1 ];
time_stop=30;
simulation_type=0;
disp('rete creata');
input('')
disp('assemblamento matrici')
[Pre,Post,M0,vel,v,D,s,alpha]=...
```

```
make_HPN...
(PreCC,PreCD,PreDC,PreDD,PostCC,...
PostCD,PostDC,PostDD,M0_C,M0_D,vel,v,D,s,alpha,it);
 disp('matrici assemblate')
 input('')

c_m=0;
ab_m=1;
disp('simulazione rete')
c=[0 1 1 1 1];
M=M0;
for k=1:max_step-1,
     [Type,M_Evol,IFS_Evol,P_macro,Event_macro_Evol,...
     firing_transition,timer_macro_event,tau,Q_Evol,timer,nc_nd_qc_qd]...
    =simulator1_HPN(Pre,Post,M0,vel,v,D,s,alpha,time_stop,simulation_type...
    ,it,c_m,ab_m,theta,c);
 m1=M_Evol(2,:);
 m2=M_Evol(3,:);
 m3=M_Evol(4,:);
 M1=M_Evol(6,:);
 M2=M_Evol(8,:);
 M3=M_Evol(10,:);
 t=tau;
 Dt=0.01;

    [dJ1c,dJ2c] = IPA(t,Dt,m1,m2,m3,M1,M2,M3,theta(1),theta(2));
    dJ1(k)=dJ1c;
    dJ2(k)=dJ2c;
    lambda=C/(k^.6);
    theta1(k+1)=theta1(k)+lambda*dJ1c;
    theta2(k+1)=theta2(k)+lambda*dJ2c;


    if theta1(k+1)>=.99 | theta1(k+1)<=0.01,
        theta1(k+1)=theta1(k);
    end
    if theta2(k+1)>=.99 | theta2(k+1)<=0.01,
       theta2(k+1)=theta2(k);
    end
    theta=[theta1(k+1) theta2(k+1)]'

% C=0.1; I am also using larger values, e.g., C=0.5 and works fine
clear m1 m2 m3 M1 M2 M3 t Dt Type M_Evol...
 IFS_Evol P_macro Event_macro_Evol firing_transition ...
```

```
      timer_macro_event tau Q_Evol theta_Evol timer nc_nd_qc_qd a b
end;
plot([0:max_step-1],theta1);
hold on
plot([0:max_step-1],theta2,'g');
```

Bibliografia

[1] F. DiCesare A. Giua. Petri Net Structural Analysis for Supervisory Control. In *IEEE Trans. on Robotics and Automation*, volume 10, pages 185–195, April 1994.

[2] Y. Wardi A. Giua, C. Seatzu. Application of IPA to fluid Petri nets. In *ADHS'09: 3nd IFAC Conf. on the Analysis and Design of Hybrid Systems*, pages 413–420, Zaragoza, Spain, September 2009.

[3] M. Fu C. Panayiotou, W. Howell. Online traffic light control through gradient estimation using stochastic fluid models. In *16th Triennial World Congress.*, Prague, Czech Republic, 2005.

[4] G. Menga F. Balduzzi, A. Giua. First-Order Hybrid Petri Nets: a Model for Optimization and Control. *IEEE Trans. on Robotics and Automation*, 16:382–399, 2000.

[5] B. Melamed G. Sun G. Panayiotou G. Cassandras, Y. Wardi. Perturbation Analysis for Online Control and Optimization of Stochastic Fluid Models. *IEEE Trans. Automatic Control*, 47:1234–1248, 2002.

[6] R.David H. Alla. Continuos and hybrid Petri nets. *J.Circuits, Syst., Comput.*, 8:159–188, 1998.

[7] D. Clark H. Kushner. *Stochastic Approximation for Constrained and Uncostrained Systems.* Springer-Verlag, 1978.

[8] J. Hespanha. Stochastic hybrid nodeling of on-off tcp flows. *CRC Press*, 34:189–217, 2006.

[9] Cao X. Ho Y. Pertutbation Analysis of Discrete Event Dynamic Systems. *Kluwer Academic Publisher*, 1991.

[10] C. Desoer L. Zadeh. *Linear Systems Theory: The State Space Approach.* McGraw Hill Series in System Science, New York,NY, 1963.

[11] M. Zazanis R. Suri P. Heidelberg, X. Cao. Convergence properties of infinitesimal perturbation analysis estimates. *Management Science*, 34:1281–1302, 1988.

[12] C.A. Petri. *Kommunication mit Automaten*. PhD thesis, Institut fur Instrumentelle Mathematik, Schriften des IIM, No. 3, Bonn, Germany, 1962.

[13] T. Pilloni C. Seatzu R. Armosini, A. Giua. Simulation and Control of a Bottling Plant using First-Order Hybrid Petri Nets. In *Proc. First Multidisciplinary Int. Symp. on Positive Systems: Theory and Applications*, volume 294, pages 79–86, Roma, Italy, 2003.

[14] H. Alla R. David. *Discrete, continous and hybrid Petri nets*. Springer, 2005.

[15] F. Sessego. HYPENS:un simulatore per le reti di Petri discrete, continue ed ibride. Master's thesis, Dep. Electric and Electronic Engineering, University of Cagliari, Cagliari, Italy, 2007. (In Italian).

[16] C. Cassandras Y. Ho, X. Cao. Infinitesimal and finite perturbation analysis for queueing networks. *Automatica*, 19:439–445, 1983.

[17] C. Cassandras C. Panayiotou Y. Wardi, B. Melamed. Ipa gradient estimators in single-node stochastic fluid models. *Journal of Optimization Theory and Applications*, 115:369–406, 2002.