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

Un toolbox per la diagnosticabilità di reti posto/transizione

Marco Pocci

Relatore: Alessandro Giua

Correlatrice: Maria Paola Cabasino

ANNO ACCADEMICO 2008-2009

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

Un toolbox per la diagnosticabilità di reti posto/transizione

Marco Pocci

Relatore: Alessandro Giua

Correlatrice: Maria Paola Cabasino

ANNO ACCADEMICO 2008-2009

Dedicato ai miei genitori e alla mia ragazza

Ringraziamenti

Vorrei ringraziare tutte quelle persone che in questi cinque anni mi hanno incoraggiato, supportato e, in alcuni casi, sopportato, permettendomi di raggiungere questo importante traguardo.

Desidero, innanzitutto, ringraziare professor Giua per i preziosi insegnamenti ricevuti durante i miei cinque anni di studio e le per le numerose ore dedicate alla mia tesi. Inoltre, ringrazio sentitamente Maria Paola Cabasino, che è stata sempre disponibile a chiarire i miei dubbi e mi ha seguito costantemente durante la stesura di questo lavoro.

Inoltre, vorrei esprimere la mia sincera gratitudine ai miei compagni di laboratorio, in particolare Andrea, Giuseppe, Stefano Salis e Stefano Scodina, per i numerosi consigli e per aver reso più piacevoli le ore trascorse in dipartimento.

Ho desiderio di ringraziare con affetto i miei genitori per il totale sostegno economico e, più di ogni altra cosa, per il grande aiuto che mi hanno dato, Veniero e Rosa, che mi hanno accolto come un figlio nei primi due anni della laurea triennale e che mi hanno accompagnato durante tutto il mio percorso formativo, Giovanni, che mi ha aiutato e ben indirizzato nei primi passi di questo lungo cammino, e, in particolar modo, Ciandra, la mia ragazza, per essermi stata accanto in questi cinque anni di lavoro, in special modo nelle avversità, e per avermi dato sempre il necessario sostegno morale.

Per ultimo, ma certamente non per importanza, vorrei ringraziare Valentino ed Alex, che sono stati sempre presenti e mi hanno permesso di affrontare con la giusta spensieratezza questi anni spesso duri.

Indice

Capitolo 1

Introduzione

La rilevazione automatica di guasti e la diagnosi si identificano in un'area di ricerca che ha avuto un discreto interesse nel corso degli ultimi anni nell'ambito dei *Sistemi ad Eventi Discreti* (SED). In questa tesi verrà, appunto, presentato un lavoro di ricerca sul problema di diagnosi e di diagnosticabilità di guasti in sistemi modellabili con una particolare classe dei SED, le *reti di Petri* (RdP). Verrà presentato un toolbox MATLAB, comprendente un insieme di programmi, scritto per la risoluzione del problema per sistemi di dimensione significativa. Nel seguito, si cercherà di inquadrare il problema della diagnosi, congiuntamente a quello della diagnosticabilità di guasto, in un contesto generale e di presentare la trattazione degli stessi argomenti con le reti posto/transizione (P/T) nel particolare. Verrà mostrata quale sia l'importanza di tali problemi e quali possano, dunque, essere i loro campi applicativi.

1.1 Problema della diagnosi

La sempre più spiccata necessità di affidabilità, sicurezza e qualità nei processi industriali sta spingendo verso un uso massiccio di sistemi automatici di misura e controllo caratterizzati da capacità di auto-diagnosi. Ciò si traduce nella realizzazione di opportune soluzioni hardware e, nel nostro caso, software per il rilievo e la diagnosi di guasti in sensori e strumenti impiegati in stazioni automatiche di misura e controllo.

La diagnosi (dal greco *dià*, attraverso e *gnosis*, conoscenza) è, appunto, il pro-

cesso atto all'identificazione di una condizione o di un fenomeno ed al suo riconducimento ad una categoria, dopo averne considerato ogni aspetto. In ambiente industriale, la diagnosi ha l'obiettivo quello di garantire la sicurezza dell'impianto e, possibilmente, anche un livello soddisfacente in termini di affidabilità delle prestazioni.

Sono molteplici le cause che possono implicare guasti in un sistema industriale e tra i più frequenti ricorrono errori nel progetto, apparecchiatura mal funzionante, sbagli dell'operatore, e così via. I tre principali fattori che spingono allo studio del problema della diagnosi dei guasti sono i seguenti:

(i) i guasti sono inevitabili;

(ii) la diagnosi dei guasti è importante (se non cruciale);

(iii) la diagnosi dei guasti è difficile.

Dal punto di vista dell'implementazione, il problema diagnostico è suddivisibile tra *problema della diagnosi* e *problema della diagnosticabilità*. In aggiunta, è possibile classificare l'approccio di analisi tra:

(i) metodi off-line;

(ii) metodi off-line.

I metodi off-line assumono che il sistema sia in un banco di prova e debba essere testato per i possibili guasti a priori, mentre nei metodi on-line il sistema si assume che sia operativo e il sottosistema diagnostico è progettato per monitorare continuamente il comportamento del sistema, identificare ed isolare i guasti. Il modello di cui faremo utilizzo è quello dei sistemi ad eventi discreti (SED), un approccio alla diagnosi appropriato per guasti notevoli o improvvisi che portano significativi cambiamenti allo stato dei componenti del sistema, ma non necessariamente portano il sistema alla rottura.

1.2 Problema della diagnosticabilità

Congiuntamente al problema di diagnosi, verrà trattato quello della diagnosticabilità, che consiste nel determinare a prori se un sistema sia diagnosticabile o no,

ossia se sia possibile ricostruire l'occorrenza di eventi di guasto osservando parole di lunghezza finita. Ci occuperemo dei sistemi di RdP limitate, seguendo il lavoro svolto da Cabasino *et al.* in [8], riportando una condizione necessaria e sufficiente per la diagnosticabilità e dando un metodo sistematico per analizzare la diagnosticabilità di una data rete di RdP. Il loro metodo richiede la costruzione di due grafi orientati ed etichettati, indicati rispettivamente come *Grafo di Raggiungibilità di Base Modificato* (MBRG) e *Diagnosticatore di Raggiungibilità di Base* (BRD), dove l'MBRG è una leggera variazione del BRG. Fondamentalmente, l'analisi consisterà nel determinare se certi cicli esistano nel BRD e, in caso affermativo, nel verificare se certe altre condizioni siano soddisfatte nell'MBRG, dunque verificare che tali cicli siano *indeterminati* o no.

Sarà obiettivo principale della tesi quello di fornire un valido strumento di diagnosi e diagnosticabilità di eventi di guasto, che faccia uso di uno schema snello improntato ai SED, nel particolare alle reti P/T. Ci occuperemo, dunque, della stesura di un programma che effettui le analisi precedentemente introdotte.

1.3 Letteratura

Il problema della diagnosi dei guasti ha ricevuto larga attenzione nella letteratura e, in questo contesto, sono stati proposti differenti ed originali approcci teorici ([20], [10], [6], [25], [14], [17]). In questa tesi, faremo riferimento al continuo e progressivo lavoro di Cabasino, Giua e Seatzu, che hanno fornito un fondamentale apporto all'applicabilità dei SED nel settore disciplinare dell'ingegneria automatica. Alla categoria dei SED appartengono tutti i sistemi dinamici i cui stati assumano diversi valori logici o simbolici, piuttosto che numerici, ed il cui comportamento sia caratterizzato dall'occorrenza di eventi istantanei, che si verificano con un cadenzamento irregolare e non necessariamente noto. Il comportamento di tali sistemi è, appunto, descritto in termini di stati e di eventi. Tale approccio è applicabile sia ai sistemi che ricadono naturalmente in questa classe, sia ai sistemi dinamici a variabili continue che possono essere ricondotti ai SED con un più alto livello di astrazione.

Le RdP, particolare modello di SED, sono sovente utilizzate in questo contesto: la loro natura intrinsecamente distribuita, dove la nozione di stato e di azione, identificate in marcatura e transizione, è locale, è stata spesso una risorsa utile alla riduzione della complessità computazionale inerente alla risoluzione di un problema di analisi. Tra i vari contributi in questa area, facciamo richiamo al lavoro di Ushio *et al.* ([24]), Aghasaryan *et al.* ([2]), Benveniste *et al.* ([4]),

Jiroveanu e Boel ([5, 15]), Basile *et al.* ([3]), Dotoli *et al.* ([11]). Per ultimo, S. Genc e S. Lafortune in [12] risolvono il problema considerato in questa tesi, utilizzando l'approccio del diagnosticatore per SED. D'altronde, gran parte di questi approcci richiede una enumerazione esaustiva dello spazio di stato.

La fondamentale differenza tra l'approccio di diagnosi mediante le RdP, presentato in ([13, 7]), e gli approcci sopraccitati sta nel concetto di *marcatura di base*. Questo concetto ci permette di rappresentare lo spazio di raggiungibilità in maniera compatta, vale a dire che l'approccio di Cabasino *et al.* richiede di enumerare solo un sottoinsieme dello spazio di raggiungibilità. Verrà data una procedura per la determinazione dell'attuale stato di diagnosi, data l'osservazione corrente e mostrato che nel caso di sistemi di reti limitati, la gran parte della procedura possa essere trasferita offline, definendo un particolare grafo, che chiameremo *Grafo di raggiungibilità di base* (BRG).

Il problema della *diagnosticabilità* è stato esaustivamente studiato nel contesto degli automi, in cui per la prima volta la diagnosticabilità è stata formalmente definita da Sampath *et al.* in [21] ed in [22]. Il loro principale contributo consiste nel fornire condizioni necessarie e sufficienti per la diagnosticabilità e presentare un metodo di diagnosi di guasto orientato agli eventi.

Al contrario, sono stati proposti estremamente pochi risultati nel campo delle RdP. Faremo costante riferimento al lavoro di Cabasino *et al.* ([8]), che permette di effettuare la diagnosi sia nel caso di reti di RdP limitate che non limitate (si veda [9]).

1.4 Struttura della tesi

Dopo aver introdotto il lettore al problema della diagnosi e della diagnosticabilità di guasti in generale ed alla trattazione con le reti di RdP nel particolare, il lavoro è organizzato nella maniera che segue.

Il capitolo 2 è dedicato alla descrizione del modello ad eventi discreti di cui verrà fatto uso nella tesi. È il caso delle reti P/T o RdP logiche, per le quali verranno richiamate le regole che ne governano l'evoluzione e tutto il formalismo che andrà ad essere utilizzato. Una rete posto/transizione consiste in un modello logico, che non tiene conto della temporizzazione degli eventi, ma solo dell'ordine con cui essi si verificano. In particolare parleremo di abilitazione e scatto di una transizione, di marcatura di una rete e insieme di raggiungibilità della stessa e, infine,

esporremo alcune proprietà che useremo nel resto della tesi.

Nel capitolo 3 è presentato un approccio alla diagnosi di SED proposto da M.P. Cabasino, A. Giua e C. Seatzu in ([9]), basato sulle RdP etichettate. La procedura proposta è basata su precedenti risultati sulle RdP non etichettate e ci permette di considerare, in aggiunta, quegli eventi che sono indistinguibili, ossia eventi che producono un segnale di uscita che è osservabile, ma che può essere comune ad altri. Il loro approccio è basato sulla nozione di marcatura di base e di vettore di giustificazione ed è mostrato come, nel caso delle RdP limitate, la parte gravosa della procedura possa essere effettuata offline, a partire dal *BRG*. In questo capitolo, daremo una definizione delle nozioni di *Spiegazioni minime ed e-vettori*, di *sequenze di scatto*, corrispondenti a una data osservazione, basata sulla nozione di marcature di base e giustificazioni. Proporremo un algoritmo per il calcolo dell'insieme delle marcature di base, che useremo per determinare un automa deterministico, che chiameremo *Grafo di raggiungibilità delle marcature di base* o più brevemente Grafo di Raggiungibilità di base, che potrà essere usato come diagnosticatore.

Nel capitolo 4 è presentato un approccio per la risoluzione del problema di diagnosticabilità per RdP limitate, presentato da Cabasino, Giua e Seatzu in [8]. Nel particolare, daremo prima di tutto condizioni necessarie e sufficienti per la diagnosticabilità. In questo capitolo, presenteremo un metodo per testare la diagnosticabilità basato sull'analisi di due grafi che dipendono dalla struttura della rete, incluso il modello di guasto e la marcatura iniziale. Il primo grafo è chiamato *Diagnosticatore di Raggiungibilità di Base* (BRD), il secondo è chiamato *Grafo di Raggiungibilità di Base Modificato* (MBRG). Di entrambi verrà fornito un algoritmo per la costruzione. La memoria necessaria per il calcolo della diagnosticabilità risulterà maggiore di quella richiesta per la diagnosi. Infatti, in questo caso sarà necessario tenere conto non solo delle marcature di base ma anche di tutte quelle marcature raggiungibili con lo scatto di transizioni di guasto. Si noti, comunque, come il numero di marcature dell'MBRG sia pari al numero di marcature raggiungibili solo nel caso peggiore, ma come usualmente sia minore.

Il capitolo 5 chiarirà le modalità di utilizzo dei programmi appartenenti al toolbox prodotto, presenterà i formalismi lessicali adottati e mostrerà degli esempi strettamente didattici. Verrà dimostrato come tali programmi possano essere considerati un valido ed efficace strumento risolutivo dei problemi trattati.

Il capitolo 6 è dedicato all'efficienza dei programmi presentati nel capitolo 5, ossia all'analisi sperimentale della loro complessità computazionale. Verranno studiate le risorse minime necessarie (in termini di tempo di calcolo) per la risoluzione del

problema di diagnosi e di quello di diagnosticabilità.

Il capitolo 7 è il capitolo conclusivo.

In appendice A sono riportati i listati dei programmi che costituiscono il toolbox di MATLAB per la diagnosi e la diagnosticabilità da me sviluppato. Tale toolbox comprende, appunto, i programmi che implementano gli algoritmi introdotti nei capitoli 3 e 4. Tali programmi, dato in ingresso un sistema di RdP le classi di guasto e la funzione di etichettatura, determinano il *BRG*, l'*MBRG* ed il *BRD* e si occupano della determinazione dei cicli indeterminati, quindi della verifica delle condizioni necessarie e sufficienti per la diagnosticabilità. A ciascuno dei programmi atti alla costruzione di un grafo ne è associato uno che fornisca a video una lettura agevole per l'utente.

1.5 Contributi della tesi

I contributi che questa tesi ha portato alla ricerca sono esclusivamente di natura implementativa. Un primo contributo originale alla tesi consiste nello sviluppo di un toolbox MATLAB per la diagnosi di guasto mediante RdP etichettate. Viene, inoltre, presentato per la prima volta un software per l'analisi della diagnosticabilità di guasti, mediante RdP etichettate.

Capitolo 2

Reti posto/transizione

Sommario

In questo capitolo, verrà richiamato il formalismo delle reti di Petri (RdP), nel particolare delle reti posto/transizione (P/T), di cui faremo uso nello sviluppo della tesi. Per maggiori dettagli facciamo riferimento al lavoro di A. Giua *et al* ([1]), da cui è tratto questo capitolo

2.1 Introduzione alle reti di Petri

Un *Sistema ad Eventi Discreti* (SED) è un sistema dinamico i cui stati possono assumere valori logici o simbolici, piuttosto che numerici, e il cui comportamento è caratterizzato dall'occorrenza di eventi istantanei che si verificano con un cadenzamento irregolare non necessariamente noto a priori. Il comportamento di tali sistemi è descritto, appunto, in termini di stati e di eventi.

Le *RdP* sono appunto un particolare modello di (SED) che trae origine da lavoro di dottorato di Carl Adam Petri ([19]), ricercatore tedesco di Lipsia. In questo capitolo, verrà fatto uso di un modello di RdP più semplice, noto come *rete posto/transizione* (P/T), un modello logico che non tiene conto della temporizzazione degli eventi ma solo dell'ordine con cui essi si verificano.

Le reti P/T sono importanti per via dei seguenti aspetti che le caratterizzano:

- Sono un formalismo matematico ed al contempo grafico e, conseguentemente, associano ad una facile comprensione una vasta gamma di applicazioni, in termini di tecniche di analisi, mirate allo studio delle proprietà che le caratterizzano.
- Godono di una rappresentazione compatta anche per spazi di stato di dimensione elevata. Esse, infatti, non richiedono la rappresentazione esaustiva dello spazio di stato, ma, bensì, delle regole che ne governano l'evoluzione.
- Permettono la rappresentazione esplicita del concetto di concorrenza.
- Permettono la rappresentazione modulare, ossia la distinzione netta tra vari sottosistemi che compongono il sistema globale e la loro unione tramite ben definiti operatori di rete.

2.2 Definizione e struttura

Una *rete 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 rettangoli.

Definizione 2.2.1. *Una* rete P/T *è una struttura* $N = (P, T, Pre, Post)$ *dove:*

- $P = \{p_1, p_2, \ldots, p_m\}$ *è l'insieme degli* m posti*;*
- $T = \{t_1, t_2, \ldots, t_n\}$ *è l'insieme delle* n transizioni*;*
- $Pre : P \times T \longrightarrow N$*: è la* funzione di pre-incidenza*, che specifica gli archi diretti dai posti alle transizioni (detti archi "pre") e viene rappresentata mediante una matrice* $m \times n$*;*
- $Post : P \times T \longrightarrow N$*: è la* funzione di post-incidenza*, che specifica gli archi diretti dalle transizioni ai posti (detti archi "post") e viene rappresentata mediante una matrice* $m \times n$*.* ■

Si suppone che $P \cap T = \emptyset$, cioè posti e transizioni siano insiemi disgiunti e che $P \cup T \neq \emptyset$, cioè la rete sia costituita da almeno un posto o una transizione.

Le matrici Pre e $Post$ sono delle matrici di interi non negativi. Si denota con $Pre(\cdot, t)$ la colonna della matrice Pre relativa alla transizione t, e con $Pre(p, \cdot)$ la riga della matrice Pre relativa al posto p. La stessa notazione vale per la matrice

$Post$. L'informazione sulla struttura di rete contenuta nelle matrici Pre e $Post$ può essere compattata in un'unica matrice, detta di incidenza.

Definizione 2.2.2. *Data una rete $N = (P, T, Pre, Post)$, con m posti ed n transizioni, la* matrice di incidenza $C : P \times T \longrightarrow Z$ *è la matrice $m \times n$ definita come:*

$$C = Pre - Post$$

cioè il generico elemento di C vale $C(p,t) = Post(p,t) - Pre(p,t)$. ■

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

Esempio 2.2.1. In Figura 2.1 è rappresentata la rete $N = (P, T, Pre, Post)$ con un insieme dei posti pari a $P = \{p_1, p_2, p_3, p_4\}$ e un insieme delle transizioni definito come $T = \{t_1, t_2, t_3, t_4, t_5\}$ le matrici Pre e Post valgono:

$$Pre = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad Post = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

La matrice di incidenza vale:

$$C = \begin{pmatrix} 0 & -1 & 0 & 0 & 1 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Si noti che $Post(p_2, t_2) = 2$ e dunque vi sono due archi che vanno dalla transizione t_2 al posto p_2. Nella figura, invece di rappresentare i due archi è usata una notazione semplificata che consiste nel rappresentare un solo arco avente per etichetta un numero (2 in questo caso) che indica la sua molteplicità. Si noti in Figura 2.2 come, ricostruendo la RdP a partire dalla matrice di incidenza, si vadano a perdere tutte le informazioni relative ad eventuali cappi. ■

Infine, data una transizione si definiscono i seguenti sistemi di posti:

${}^\bullet t = \{p \in P \mid Pre(p,t) > 0\}$: è l'insieme dei posti in ingresso a t.
$t^\bullet = \{p \in P \mid Post(p,t) > 0\}$: è l'insieme dei posti in uscita da t.

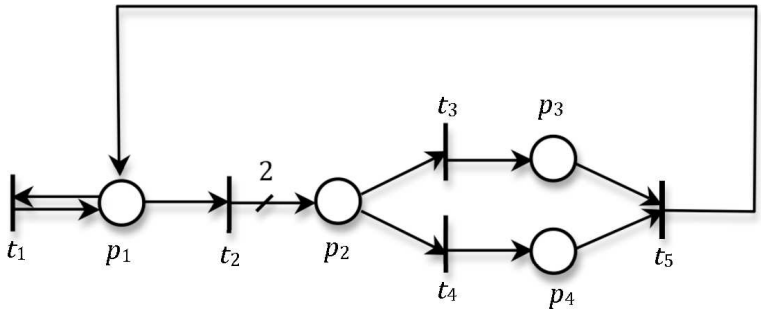

Figura 2.1: Una rete P/T

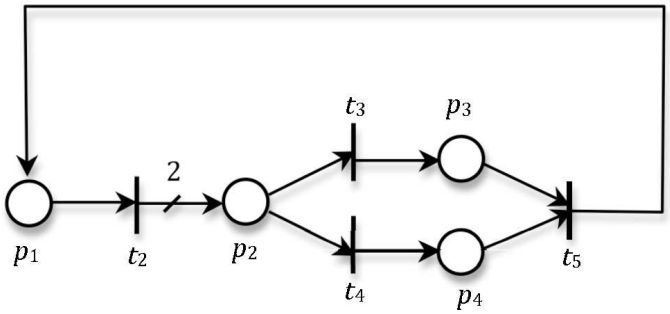

Figura 2.2: Una rete P/T ricostruita a partire dalla sua matrice di incidenza

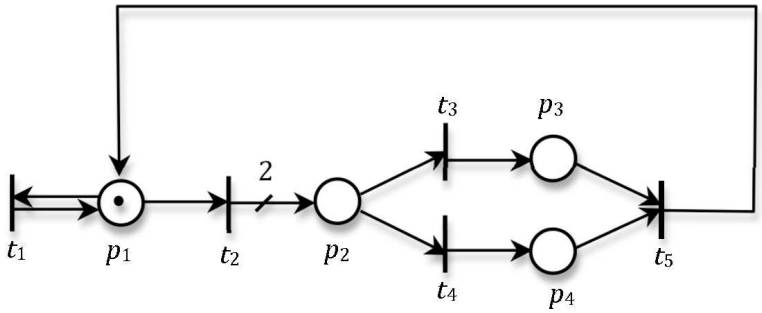

Figura 2.3: Evoluzione di una rete marcata. Marcatura iniziale.

${}^\bullet p = \{t \in T \mid Post(p,t) > 0\}$: è l'insieme delle transizioni in ingresso a p.
$p^\bullet = \{t \in T \mid Pre(p,t) > 0\}$: è l'insieme delle transizioni in uscita da p.

Ad esempio nella rete in Figura 2.1 vale ${}^\bullet t_2 = \{p_1\}$, $t_2^\bullet = \{p_2\}$, ${}^\bullet p_2 = \{t_2\}$, $p_2^\bullet = \{t_3, t_4\}$.

2.3 Marcatura e sistema di rete

Mediante la marcatura è possibile definire lo stato di una rete P/T.

Definizione 2.3.1. *Una* marcatura *è una funzione $M : P \longrightarrow N$ che assegna ad ogni posto un numero intero non negativo di marche (o gettoni) rappresentate graficamente con dei pallini neri dentro i posti.* ■

Considerando l'esempio in Figura 2.1, una marcatura possibile M è $M(p_1) = 1$, $M(p_2) = M(p_3) = M(p_4) = 0$ come mostrato in Figura 2.3. Un'altra marcatura possibile è quella mostrata in Figura 2.4, dove $M(p_1) = 0, M(p_2) = 2, M(p_3) = M(p_4) = 0$.

Definizione 2.3.2. *Una rete N con una marcatura iniziale M_0 è detta* rete marcata o sistema di rete*, e viene indicata come $\langle N, M_0 \rangle$.* ■

Una rete marcata è, in effetti, un sistema ad eventi discreti a cui è associato un comportamento dinamico.

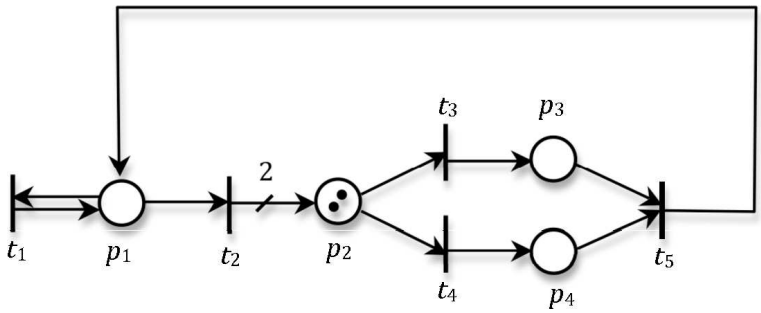

Figura 2.4: Evoluzione di una rete marcata. Marcatura raggiunta dopo lo scatto della transizione t_2.

2.4 Abilitazione e scatto

Definizione 2.4.1. *Una* transizione t *è* abilitata *dalla marcatura* M *se*

$$M \geqslant Pre(\cdot, t)$$

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* M *si scrive* $M[t\rangle$*. Per indicare che* t' *non è abilitata da* M *si scrive* $\neg M[t'\rangle$*.* ■

Definizione 2.4.2. *Una transizione* t *abilitata da una marcatura* M *può scattare. Lo scatto di* t *rimuove* $Pre(p, t)$ *marche da ogni posto* p *appartenente a* P *e aggiunge* $Post(p, t)$ *in ogni posto* p *appartenente a* P*, determinando una nuova marcatura* M'*. Cioè vale:*

$$M' = M - Pre(\cdot, t) + Post(\cdot, t) = M + C(\cdot, t)$$

Per indicare che lo scatto della transizione t *da* M *determina la marcatura* M' *si scrive* $M[t\rangle M'$*.* ■

Definizione 2.4.3. *Il comportamento (o linguaggio) di una rete marcata* $\langle N, M_0\rangle$ *è l'insieme delle sequenze di scatto abilitate dalla marcatura iniziale, cioè l'insieme:*

$$L(N, M_0) = \{\sigma \in T^* : M_0[\sigma\rangle\}.$$

■

T^* è l'insieme di tutte le possibili sequenze di transizioni facenti parte dell'insie-

me T.

Definizione 2.4.4. *Una sequenza di transizioni $\sigma = t_{j_1}t_{j_2}...t_{j_r} \in T^*$ è abilitata da una marcatura M, se:*

(i) la transizione t_{j_1} è abilitata da M e il suo scatto porta ad $M_1 = M + C(\cdot, t_{j_1})$;

(ii) la transizione t_{j_2} è abilitata da M_1 e il suo scatto porta ad $M_2 = M_1 + C(\cdot, t_{j_2})$ ecc.

Una sequenza abilitata σ viene anche detta sequenza di scatto e ad essa corrisponde la traiettoria:

$$M[t_{j_1}\rangle M_1[t_{j_2}\rangle M_2...[t_{j_r}\rangle M_r.$$

■

Per indicare che la sequenza σ è abilitata da M si scrive $M[\sigma\rangle$. Per indicare che lo scatto di σ da M determina la marcatura M' si scrive $M[\sigma\rangle M'$. Ad esempio, nella rete in Figura 2.4 una possibile sequenza di transizioni abilitata dalla marcatura data è $\sigma = t_3t_4t_5t_1$ il cui scatto porta alla marcatura iniziale $M_0 = [\ 1 \quad 0 \quad 0 \quad 0\ ]^T$.

Definizione 2.4.5. *Il comportamento (o linguaggio) di una rete marcata $\langle N, M_0\rangle$ è l'insieme delle sequenze di scatto abilitate dalla marcatura iniziale, cioè l'insieme:*

$$L(N, M_0) = \{\sigma \in T^* : M_0[\sigma\rangle\}.$$

■

Definizione 2.4.6. *Una marcatura M è detta raggiungibile in $\langle N, M_0\rangle$ se esiste una sequenza di scatto tale che $M_0[\sigma\rangle M$. L'*insieme di raggiungibilità *di una rete marcata $\langle N, M_0\rangle$ è l'insieme delle marcature che possono venir raggiunte a partire dalla marcatura iniziale, cioè l'insieme:*

$$R(N, M_0) = \{M \in N^m \mid \exists\sigma \in L(N, M_0) : M_0[\sigma\rangle M\}.$$

Definiamo insieme potenzialmente raggiungibile $PR(N, M_0)$ *, l'insieme:*

$$PR(N, M_0) = \{M \in N^m \mid \exists y \in N^n : M = M_0 + C \cdot y\}.$$

I due insiemi sono legati dalla relazione $R(N, M_0) \subseteq PR(N, M_0)$. ■

2.5 Equazione di stato

Definizione 2.5.1. *Sia $\langle N, M_0 \rangle$ una rete marcata e C sia la sua matrice di incidenza. Se M è raggiungibile da M_0 scattando la sequenza di transizioni σ vale:*

$$M = M_0 + C \cdot \sigma.$$

σ è detto vettore di scatto e ha tante componenti quante sono le transizioni. ■

2.6 Proprietà di una rete di Petri

In questo paragrafo daremo alcune proprietà che ci saranno molto utili nel capitolo 3, dove si parla dell'approccio con le RdP.

Un sistema di RdP $\langle N, M_0 \rangle$ è *vivo* per tutte le $M \in R(N, M_0)$ se esiste almeno una transizione $t \in T$ che è abilitata.

Una RdP che non ha cicli orientati è detta *aciclica*. Per questa sottoclasse di reti valgono i seguenti risultati:

Teorema 2.6.1. *Consideriamo una* rete *di Petri* aciclica N.

(i) Se il vettore $y \in \mathbb{N}^n$ soddisfa l'equazione $M_0 + C \cdot y \geq 0$, allora esiste una sequenza di scatto σ "scattabile" dalla marcatura M_0 e tale che il vettore di scatto associato a σ è uguale a y.

(ii) Una marcatura M è raggiungibile da una marcatura M_0 se e solo se esiste una soluzione intera e non negativa y che soddisfa l'equazione di stato $M = M_0 + C \cdot y$, ossia $R(N, M_0) = PR(N, M_0)$. ■

Un posto p è *limitato* se:

$$bound(p) = \max_{M \in R(N, M_0)} M(p) = k < +\infty \tag{2.1}$$

Un sistema di rete $\langle N, M_0 \rangle$ è *limitato* se esiste una costante k positiva tale che, per ogni $M \in R(N, M_0)$ e per ogni $p \in P$ è verificato che $M(p) \leq k$. Una rete è detta *strutturalmente limitata* se è limitata per qualunque marcatura iniziale.

2.7 Reti di Petri etichettate

Osservando l'evoluzione della rete, è comune assumere che a ciascuna transizione t sia assegnata un'etichetta $\mathcal{L}(t)$ e che l'occorrenza di t dia luogo ad un'uscita osservabile $\mathcal{L}(t)$. Se la transizione fosse etichettata con la parola vuota, $\mathcal{L}(t) = \varepsilon$, il suo scatto non potrebbe essere osservato. Questo porta alla definizione di reti etichettate.

Definizione 2.7.1. *Data una RdP N con un insieme di transizioni T, una* funzione di etichettatura $\mathcal{L} : T \to E \cup \{\varepsilon\}$ *assegna a ciascuna transizione $t \in T$ un simbolo, da un dato alfabeto E, o la parola vuota ε.*

Un sistema di RdP etichettato *è una tripla $G = \langle N, M_0, \mathcal{L} \rangle$ dove $N = (P, T, Pre, Post)$, M_0 è la marcatura iniziale e $\mathcal{L} : T \to E \cup \{\varepsilon\}$ è la funzione di etichettatura.* ■

Quattro classi di funzione di etichettatura possono essere definite.

Definizione 2.7.2. *La funzione di etichettatura di un sistema di RdP etichettato $\langle N, M_0, \mathcal{L} \rangle$ può essere classificato come segue.*

- Free*: se tutte le transizioni sono etichettate distintamente, ossia se a ciascuna transizione è associata una etichetta distinta e nessuna transizione è etichettata con la parola vuota.*

- Deterministica*: se nessuna transizione è etichettata con la parola vuota ed è sempre verificato che due transizioni abilitate simultaneamente non possano condividere la stessa etichetta. Più formalmente, devono valere le seguenti condizioni: per tutte le $t, t' \in T$, con $t \neq t'$, e per tutte le $M \in R(N, M_0)$: $M[t\rangle \wedge M[t'\rangle \Rightarrow [\mathcal{L}(t) \neq \mathcal{L}(t')]$. Ciò assicura che la conoscenza dello scatto di etichette $\mathcal{L}(t)$ sia sufficiente per ricostruire la marcatura a cui lo scatto di t porta.*

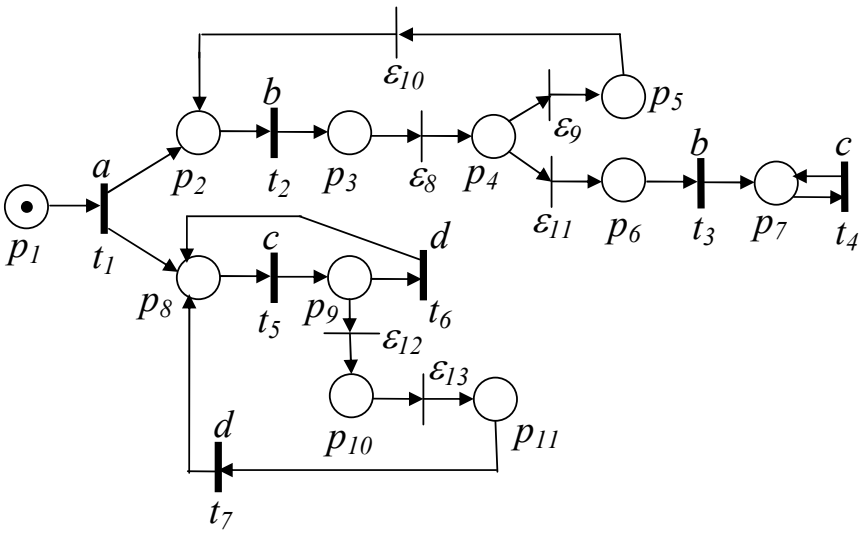

Figura 2.5: Un esempio di RdP etichettata

- λ-free*: se nessuna transizione è etichettata con la parola vuota*[1].
- Arbitraria*: se la funzione di etichettatura $\mathcal{L}$ non è vincolata ad alcuna restrizione.* ■

Ciascuno di questi tipi di funzioni di etichettatura è una generalizzazione del precedente caso. In aggiunta, tutti i tipi dipendono solo dalla struttura della rete, fatta eccezione per l'etichettatura deterministica, che dipende anche dal comportamento della rete.

Nel caso specifico in cui la funzione di etichettatura sia di tipo *free*, essendo un isomorfismo tra l'alfabeto E e l'insieme di transizioni T, si è soliti scegliere $E = T$, o, equivalentemente, assumere che le transizioni non siano etichettate e che il loro scatto sia direttamente osservabile.

Indichiamo con T_u l'insieme delle transizioni la cui etichetta è ε, ossia $T_u = \{t \in T \mid \mathcal{L}(t) = \varepsilon\}$. Le transizioni di T_u sono dette *non osservabili* o *silenziose*. Indichiamo con T_o l'insieme delle transizioni etichettate con un simbolo in $\mathcal{L}$. Le transizioni di T_o sono dette *osservabili*, poichè quando scattano possono essere osservate.

[1]Nella letteratura delle RdP, la parola vuota è indicata con λ, mentre nella letteratura del linguaggio formale è indicata con ε. In questa tesi indicheremo la parola vuota con ε ma, per consistenza con la letteratura delle RdP, useremo il termine λ*-free*, se nessuna transizione è etichettabile con la parola vuota.

Si tenga presente che in questa tesi assumiamo che la stessa etichetta $l \in \mathcal{L}$ possa essere associata a più di una transizione. In particolare, due transizioni $t_1, t_2 \in T_o$ sono dette *indistinguibili* se condividono la stessa etichetta, ossia $\mathcal{L}(t_1) = \mathcal{L}(t_2) = l \in \mathcal{L}$. L'insieme delle transizioni che condividono la stessa etichetta l è indicato con T_l.

Nel seguito indicheremo con C_u (C_o) la restrizione della matrice di incidenza a T_u (T_o). Indichiamo con w la stringa di eventi associata alla sequenza σ, tale che $w = \mathcal{L}(\sigma)$. Notiamo che la lunghezza di una sequenza σ (indicata con $|\sigma|$) è sempre maggiore o uguale della lunghezza della corrispondente parola w (indicata con $|w|$). Infatti, se σ contiene k' transizioni etichettate con ε allora $|\sigma| = k' + |w|$.

Definizione 2.7.3. *Data una rete $N = (P, T, Pre, Post)$ e un sottoinsieme $T' \subseteq T$ delle sue transizioni, definiamo la sottorete $T'-$indotta di N la nuova rete $N' = (P, T', Pre', Post')$, dove $Pre', Post'$ sono le restrizioni di Pre e $Post$ a T'. La nuova rete N' può essere pensata come ottenuta da N rimuovendo tutte le transizioni $T \setminus T'$. Scriveremo anche $N' \prec_{T'} N$.* ■

Nel seguito faremo uso della notazione $\mathcal{L}^{-1}(l)$ per indicare l'insieme di transizioni la cui etichetta è pari a l. Analogamente, data una sequenza di etichette osservate $w \in E^*$, indichiamo con $\mathcal{L}^{-1}(w)$ l'insieme delle sequenze di transizioni $\sigma \in T^* \mid \mathcal{L}(\sigma) = w$.

In aggiunta, indichiamo con σ_0 la sequenza di lunghezza nulla e ε la parola vuota. Utilizziamo la notazione $w_i \preccurlyeq w$ per indicare il generico prefisso w di lunghezza $i \leq k$, dove k è la lunghezza di w. In particolare, per $i = 0$, abbiamo per definizione la parola vuota, $w_0 = \varepsilon$.

Definiamo l'*operatore di proiezione* $P : T^* \to T_o^*$ ricorsivamente come segue:

(i) $P(t_j) = t_j, \ \forall t_j \in T_o$;

(ii) $P(t_i) = \varepsilon, \ \forall t_i \in T_u$;

(iii) $P(\sigma t_j) = P(\sigma)P(t_j), \ \forall \sigma \in T^*, t_j \in T$.

Inoltre, data una sequenza $\sigma \in T^*$, chiamiamo $\pi : T^* \to \mathbb{N}^n$ La funzione che associa a σ un vettore $y \in \mathbb{N}^n$, noto come il *vettore di scatto* di σ. In particolare, $y = \pi(\sigma)$ è tale che $y(t) = k$, se a transizione t è contenuto k volte in σ.

Capitolo 3

Diagnosi mediante reti di Petri

Sommario

Questo capitolo espone un approccio alla diagnosi di sistemi ad eventi discreti (SED) mediante le reti di Petri (RdP) etichettate, presentato in [9] da M.P. Cabasino, A. Giua e C. Seatzu. La procedura proposta è basata su precedenti risultati ottenuti sulle RdP non etichettate e permette di considerare, in aggiunta, quegli eventi che sono indistinguibili, ossia gli eventi che producono un segnale di uscita che è osservabile, ma che è comune ad altri. L'approccio è basato sulla nozione di marcatura di base e di vettore di giustificazione ed è mostrato come, nel caso delle RdP limitate, la parte gravosa della procedura possa essere effettuata offline, a partire da un particolare grafo, che noi chiameremo *Grafo di raggiungibilità di base* (BRG).

3.1 Marcature e sequenze di scatto consistenti ad una data osservazione

Definizione 3.1.1. *Sia $\langle N, M_0 \rangle$ un sistema di RdP etichettata con funzione di etichettatura $\mathcal{L} : T \to L \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Sia $w \in L^*$ una parola osservata.*

Definiamo

$$\mathcal{S}(w) = \{\sigma \in L(N, M_0) \mid \mathcal{L}(\sigma) = w\}$$

l'insieme delle sequenze di scatto consistenti *con* $w \in L^*$ *e*

$$\mathcal{C}(w) = \{M \in R(N, M_0) \mid \exists \sigma \in T^* : \mathcal{L}(\sigma) = w \wedge M_0[\sigma\rangle M\}$$

l'insieme delle marcature consistenti *con* $w \in L^*$. ■

In parole povere, data una certa osservazione w, $\mathcal{S}(w)$ è l'insieme delle sequenze che possono essere scattate, essendo $\mathcal{C}(w)$ l'insieme delle marcature in cui il sistema potrebbe attualmente essere.

Esempio 3.1.1. Consideriamo la RdP in Figura 2.5. Assumiamo che $T_o = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ e $T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}\}$, dove, per una migliore comprensione, le transizioni non osservabili vengono indicate con ε_i anzichè con t_i. La funzione di etichettatura è definita come segue: $\mathcal{L}(t_1) = a$, $\mathcal{L}(t_2) = \mathcal{L}(t_3) = b$, $\mathcal{L}(t_4) = \mathcal{L}(t_5) = c$, $\mathcal{L}(t_6) = \mathcal{L}(t_7) = d$.

Innanzitutto, consideriamo $w = acd$. L'insieme delle sequenze di scatto consistenti con w è $\mathcal{S}(w) = \{t_1t_5t_6, t_1t_5\varepsilon_{12}\varepsilon_{13}t_7\}$, mentre l'insieme delle marcature consistenti con w è $\mathcal{C}(w) = \{[0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T\}$. Dunque, possono essere scattate due distinte sequenze di scatto (la seconda comprende anche transizioni silenziose), ma entrambe portano alla stessa marcatura.

Possono essere raggiunte marcature distinte se consideriamo $w = ab$. In particolare, si ha che $\mathcal{S}(w) = \{t_1t_2, t_1t_2\varepsilon_8, t_1t_2\varepsilon_8\varepsilon_9, t_1t_2\varepsilon_8\varepsilon_9\varepsilon_{10}, t_1t_2\ \varepsilon_8\varepsilon_{11}\}$, mentre $\mathcal{C}(w) = \{[0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T, [0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T, [0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0]^T, [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T, [0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0]^T\}$. ■

3.2 Spiegazioni minime ed e-vettori minimi

In [7] sono state date le seguenti due definizioni.

Definizione 3.2.1. Data una marcatura M ed una transizione osservabile $t \in T_o$, definiamo

$$\Sigma(M, t) = \{\sigma \in T_u^* \mid M[\sigma\rangle M',\ M' \geq Pre(\cdot, t)\}$$

l'insieme delle *spiegazioni* di t ad M, e

$$Y(M, t) = \pi(\Sigma(M, t))$$

gli *e-vettori* (o *vettori di spiegazione*), ossia i vettori di scatto associati alle spiegazioni. ■

Conseguentemente $\Sigma(M,t)$ è l'insieme delle sequenze non osservabili il cui scatto a M abilita t. Tra le precedenti sequenze, vogliamo selezionare quelle il cui vettore di scatto è minimo.

Definizione 3.2.2. Data una marcatura M ed una transizione $t \in T_o$, definiamo

$$\Sigma_{\min}(M,t) = \{\sigma \in \Sigma(M,t) \mid \nexists\, \sigma' \in \Sigma(M,t) : \pi(\sigma') \lneqq \pi(\sigma)\}$$

l'insieme delle *spiegazioni minime* di t in M e definiamo

$$Y_{\min}(M,t) = \pi(\Sigma_{\min}(M,t))$$

il corrispondente insieme di *e-vettori minimi*, vettore di scatto associato a tali sequenze. ■

In [9] vengono generalizzate le condizioni di cui sopra.

Definizione 3.2.3. *Data una marcatura M ed una osservazione $l \in L$, definiamo l'insieme delle* spiegazioni minime di l a partire da M *come*

$$\hat{\Sigma}_{\min}(M,l) = \cup_{t \in T_l} \cup_{\sigma \in \Sigma_{\min}(M,t)} (t,\sigma),$$

ossia l'insieme delle coppie (transizione etichettata l; spiegazione minima corrispondente), e definiamo l'insieme degli e-vettori minimi *di l a partire da M come*

$$\hat{Y}_{\min}(M,l) = \cup_{t \in T_l} \cup_{e \in Y_{\min}(M,t)} (t,e),$$

ossia l'insieme delle coppie (transizione etichettata l; e-vettore minimo corrispondente). ■

Ovviamente, $\hat{\Sigma}_{\min}(M,l)$ e $\hat{Y}_{\min}(M,l)$ sono una generalizzazione dell'insieme delle spiegazioni minime e degli e-vettori minimi introdotti per RdP non etichettate con transizioni non osservabili. D'altronde, nei precedenti insiemi $\hat{\Sigma}_{\min}(M,l)$ e $\hat{Y}_{\min}(M,l)$, differenti sequenze σ e differenti e-vettori e, rispettivamente, sono associati in generale alla stessa $t \in T_l$.

3.3 Marcature di base e g-vettori

In [7] la nozione di *marcatura di base* e *vettore di giustificazione g-vettore* è stata definita per RdP non etichettate. In particolare, data una sequenza di transizioni $w \in T_o^*$, una marcatura di base M_b è una marcatura raggiunta da M_0 con lo scatto

della parola osservata w e di tutte le transizioni non osservabili il cui scatto è necessario per abilitare w. Un g-vettore $y \in Y_{min}(M_0, w)$ è un vettore di scatto di transizioni non osservabili il cui scatto è necessario per raggiungere M_b.
Qui fondamentalmente facciamo uso delle stesse definizioni, benchè con una piccola ma significativa differenza. Infatti, nel caso di RdP etichettate, l'osservazione w è una sequenza di etichette, ossia $w \in L^*$. In generale, diverse sequenze $\sigma_o \in T_o^*$ possono corrispondere alla stessa w, ossia varie sequenze di transizioni osservabili, tali che $\mathcal{L}(\sigma_o) = w$, possono attualmente essere scattate. In generale, per l'abilitazione di ciascuna di queste sequenze σ_o è necessaria una specifica sequenza di transizioni non osservabili, intervallata alle stesse.

Dunque, risulta necessario introdurre la seguente definizione di coppie (sequenza di transizioni in T_o etichettata w; corrispondente *giustificazione*).

Definizione 3.3.1. Sia $\langle N, M_0\rangle$ un sistema di rete con funzione di etichettatura $\mathcal{L} : T \to L \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Sia $w \in L^*$ una data osservazione. Definiamo

$$\begin{aligned}\hat{\mathcal{J}}(w) \;= \{\, &(\sigma_o, \sigma_u),\ \sigma_o \in T_o^*,\ \mathcal{L}(\sigma_o) = w,\ \sigma_u \in T_u^* \mid \\ &[\exists \sigma \in \mathcal{S}(w) \;:\; \sigma_o = P_o(\sigma),\ \sigma_u = P_u(\sigma)] \wedge \\ &[\nexists \sigma' \in \mathcal{S}(w) :\ \sigma_o = P_o(\sigma'),\ \sigma'_u = P_u(\sigma') \wedge\ \pi(\sigma'_u) \lneq \pi(\sigma_u)]\}\end{aligned}$$

l'insieme di coppie (sequenza $\sigma_o \in T_o^*$ con $\mathcal{L}(\sigma_o) = w$; corrispondente *giustificazione* di w). In aggiunta, definiamo

$$\begin{aligned}\hat{Y}_{\min}(M_0, w) = \{(\sigma_o, y),\ \ &\sigma_o \in T_o^*,\ \mathcal{L}(\sigma_o) = w,\ y \in \mathbb{N}^{n_u} \mid \\ &\exists (\sigma_o, \sigma_u) \in \hat{\mathcal{J}}(w) : \pi(\sigma_u) = y\}\end{aligned}$$

l'insieme delle coppie (sequenza $\sigma_o \in T_o^*$ con $\mathcal{L}(\sigma_o) = w$; corrispondente *g-vettore*). ■

In altre parole, $\hat{\mathcal{J}}(w)$ è l'insieme delle coppie il cui primo elemento è la sequenza $\sigma_o \in T_o^*$ etichettata w ed il cui secondo elemento è la corrispondente sequenza di transizioni non osservabili intervallate con σ_o, il cui scatto abilita σ_o ed il cui vettore di scatto è minimo. Il vettore di scatto di tali sequenze è detto *g-vettore*.

Definizione 3.3.2. Sia $\langle N, M_0\rangle$ un sistema di rete con funzione di etichettatura $\mathcal{L} : T \to L \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Sia w una data osservazione e $(\sigma_o, \sigma_u) \in \hat{\mathcal{J}}(w)$ una coppia generica (sequenza etichettata di transizioni osservabili w – giustificazione minima corrispondente). La marcatura

$$M_b = M_0 + C_u \cdot y + C_o \cdot y', \qquad y = \pi(\sigma_u), \qquad y' = \pi(\sigma_o),$$

ossia la marcatura raggiunta una volta scattata σ_o, intervallata con la giustificazione minima σ_u, è detta *marcatura di base* e y viene detto suo *g-vettore* (o *vettore di giustificazione*). ■

Ovviamente, poiché, in generale, per una parola $w$esiste più di una giustificazione (la cardinalità dell'insieme $\hat{\mathcal{J}}(w)$ è generalmente maggiore di uno), la marcatura di base può non essere unica.

Definizione 3.3.3. Sia $\langle N, M_0\rangle$ un sistema di rete con funzione di etichettatura $\mathcal{L} : T \to L \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Sia $w \in L^*$ una parola osservata. Definiamo

$$\mathcal{M}(w) = \{(M, y) \mid \ (\exists \sigma \in \mathcal{S}(w) \ : \ M_0[\sigma\rangle M) \ \wedge (\exists (\sigma_o, \sigma_u) \in \hat{\mathcal{J}}(w) \ : \\ \sigma_o = P_o(\sigma), \qquad \sigma_u = P_u(\sigma), \ y = \pi(\sigma_u))\}$$

come l'insieme delle coppie (marcature di base; relativo g-vettore) che sono *consistenti* con $w \in L^*$. ■

Si noti che l'insieme $\mathcal{M}(w)$ non tiene conto delle sequenze di transizioni osservabili che possono essere scattate attualmente, bensì tiene traccia solo delle marcature che possono essere raggiunte e del vettore di scatto relativo a sequenze di transizioni non osservabili, che sono scattate per raggiungerle. Di fatto, questa è l'informazione maggiormente significativa nell'effettuare la diagnosi. La nozione di $\mathcal{M}(w)$ è fondamentale per ottenere uno strumento ricorsivo per determinare l'insieme delle spiegazioni minime.

Proposizione 3.3.1. *Dato un sistema di rete $\langle N, M_0\rangle$ con una funzione di etichettatura $\mathcal{L} : T \to L \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Assumiamo che la sottorete T_u-indotta sia aciclica. Sia $w = w'l$ una data osservazione.*

L'insieme $\hat{Y}_{\min}(M_0, wl)$ è definito come:

$$\hat{Y}_{\min}(M_0, wl) = \ \{(\sigma_o, y) \mid \ \sigma_o = \sigma_0' t \wedge y = y' + e \ : \\ (\sigma_o', y') \in \hat{Y}_{\min}(M_0, w), (t, e) \in \hat{Y}_{\min}(M_b', l) \text{ and } \mathcal{L}(t) = l\},$$

dove $M_b' = M_0 + C_u \cdot y' + C_o \cdot \sigma_o'$. □

Esempio 3.3.1. Consideriamo la RdP in Figura 2.5 precedentemente introdotta nell'Esempio 3.1.1. Assumiamo $w = acd$. L'insieme delle giustificazioni è $\hat{\mathcal{J}}(w) = \{(t_1t_5t_6, \varepsilon), (t_1t_5t_7, \ \varepsilon_{12}\varepsilon_{13})\}$ e l'insieme dei g-vettori è $\hat{Y}_{min}(M_0, w) = \{(t_1t_5t_6, \vec{0}),$
$(t_1t_5t_7, [0\,0\,0\,0\,1\,1]^T)\}$. I precedenti g-vettori portano alla stessa marcatura di base $M_b = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$, dunque $\mathcal{M}(w) = \{(M_b, \vec{0}), (M_b, [0\,0\,0\,0\,1\,1]^T)\}$.

Ora, consideriamo $w = ab$. In questo caso $\hat{\mathcal{J}}(w) = \{(t_1 t_2, \varepsilon)\}$, $\hat{Y}_{min}(M_0, w) = \{(t_1 t_2, \vec{0})\}$ e la marcatura di base è la stessa del caso precedente, ossia $M_b = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$, dunque $\mathcal{M}(w) = \{(M_b, \vec{0})\}$. ■

Sotto l'assunzione di aciclicità della sottorete T_u-indotta, l'insieme $\mathcal{M}(w)$ può facilmente essere determinato come segue.

Algoritmo 3.3.1. [Calcolo delle marcature di base e dei g-vettori]

1. Sia $w = \varepsilon$.
2. Sia $\mathcal{M}(w) = \{(M_0, \vec{0})\}$.
3. Attendi sino a che una nuova etichetta l non viene osservata.
4. Sia $w' = w$ e $w = w'l$.
5. Sia $\mathcal{M}(w) = \emptyset$.
6. Per tutte le M' tali che $(M', y') \in \mathcal{M}(w')$, fai
  6.1. per tutte le $t \in T_l$, fai
    6.1.1. per tutte le $e \in Y_{\min}(M', t)$, fai
    6.1.1.1. sia $M = M' + C_u \cdot e + C(\cdot, t)$,
    6.1.1.2. per tutte le y' tali che $(M', y') \in \mathcal{M}(w')$, fai
      6.1.2.1. sia $y = y' + e$,
      6.1.2.2. sia $\mathcal{M}(w) = \mathcal{M}(w) \cup \{(M, y)\}$.
7. Ritorna al passo 3. ■

In altre parole, l'algoritmo di cui sopra può essere spiegato come segue.

Assumiamo che inizialmente non sia stata osservata alcuna parola e che, quindi, il sistema di RdP si trovi alla sua marcatura iniziale M_0, con associata la giustificazione nulla (essendo nullo il g-vettore associato alla parola vuota). Sia dunque $\mathcal{M}(w) = \{(M_0, \vec{0})\}$. In seguito, una nuova transizione osservabile t scatta e viene osservata la sua etichetta $\mathcal{L}(t)$ (e.g., l). Consideriamo tutte le marcature di base all'osservazione w' prima dello scatto di t, selezioniamo tra queste quelle che possono aver permesso lo scatto di almeno una transizione $t \in T_l$. Si tenga sempre conto che questo può aver richiesto lo scatto di un'appropriata sequenza di transizioni non osservabili. In particolare, focalizziamo la nostra attenzione sulle spiegazioni minime e sui corrispondenti e-vettori (passo 6.1.1). Per ultimo, aggiorniamo l'insieme $\mathcal{M}(w)$, includendo tutte le coppie di nuove marcature di base e g-vettori, tenendo conto che, per ciascuna marcatura di base, raggiunta con l'osservazione di w', può corrispondere più di un g-vettore.

Richiamiamo, ora, i due risultati che seguono da [7].

Definizione 3.3.4. *Sia $\langle N, M_0 \rangle$ un sistema di rete $N = (P, T, Pre, Post)$ e $T =$*

$T_o \cup T_u$. Si assuma che la sottorete T_u-indotta sia aciclica. Sia $w \in T_o^*$ una parola osservata. Indichiamo con

$$\mathcal{M}_{basis}(w) = \{M \in \mathbb{N}^m \mid \exists y \in \mathbb{N}^{n_u} \text{ e } (M, y) \in \mathcal{M}(w)\}$$

l'insieme delle marcature di base ottenute in seguito all'osservazione di w. Inoltre, indichiamo con

$$\mathcal{M}_{basis} = \bigcup_{w \in T_o^*} \mathcal{M}_{basis}(w)$$

l'insieme di tutte le marcature di base ottenute in seguito ad ogni osservazione $w \in T_0^*$. ■

Si noti che, se la rete è limitata, allora l'insieme $\mathcal{M}_{basis}$ è *finito*, essendo l'insieme delle marcature di base un sottoinsieme dell'insieme di raggiungibilità.

Teorema 3.3.1. *Consideriamo un sistema di rete $\langle N, M_0 \rangle$ la cui sottorete non osservabile sia aciclica. Per ciascuna $w \in L^*$ si ha che*

$$\mathcal{C}(w) = \{M \in \mathbb{N}^m \mid M = M_b + C_u \cdot y \ : y \geq \vec{0} \text{ and } M_b \in \mathcal{M}_{basis}(w)\}.$$

□

3.4 Diagnosi utilizzando le reti di Petri

Si assuma che l'insieme delle transizioni non osservabili sia partizionato in due sottoinsiemi, cioé $T_u = T_f \cup T_{reg}$ dove T_f include tutte le transizioni di guasto (che modellano comportamenti anomali o di guasto), mentre T_{reg} include tutte le transizioni relative ad eventi non osservabili ma regolari. L'insieme T_f è ulteriormente partizionato in r distinti sottoinsiemi T_f^i, dove $i = 1, \ldots, r$, che modellano le differenti classi di guasto.

La seguente definizione introduce la nozione di *diagnosticatore* ed è basata su ciò che è stato introdotto in [7], nel caso di RdP non etichettate.

Definizione 3.4.1. *Un* diagnosticatore *è una funzione $\Delta : L^* \times \{T_f^1, T_f^2, \ldots, T_f^r\} \rightarrow \{0, 1, 2, 3\}$ che associa a ciascuna osservazione $w \in L^*$ ed a ciascuna classe di guasto T_f^i, $i = 1, \ldots, r$, uno* stato di diagnosi.

- $\Delta(w, T_f^i) = 0$ *se per tutte le $\sigma \in \mathcal{S}(w)$ e per tutte le $t_f \in T_f^i$ è verificato che $t_f \notin \sigma$.*

In questo caso, non si è verificato alcun guasto associato alla i-esima classe di guasto, in quanto nessuna delle sequenze di scatto consistente con l'osservazione contiene una transizione di guasto della i-esima classe,

- $\Delta(w, T_f^i) = 1$ *se:*

  (i) esiste una $\sigma \in \mathcal{S}(w)$ e una $t_f \in T_f^i$ tali che $t_f \in \sigma$ ma

  (ii) per tutte le $(\sigma_o, \sigma_u) \in \hat{\mathcal{J}}(w)$ e per tutte le $t_f \in T_f^i$ si ha che $t_f \notin \sigma_u$.

  In questo caso una transizione di guasto della classe i-esima può essersi verificata ma non è contenuta in nessuna giustificazione di w.

- $\Delta(w, T_f^i) = 2$ *se esistono due coppie $(\sigma_o, \sigma_u), (\sigma'_o, \sigma'_u) \in \hat{\mathcal{J}}(w)$ tali che:*

  (i) esiste $t_f \in T_f^i$ tale che $t_f \in \sigma_u$;

  (ii) per tutte le $t_f \in T_f^i$, $t_f \notin \sigma'_u$.

  In questo caso una transizione di guasto della classe i è contenuta in una (ma non in tutte) giustificazione di w.

- $\Delta(w, T_f^i) = 3$ *se per tutte le $\sigma \in \mathcal{S}(w)$ esiste una $t_f \in T_f^i$ tale che $t_f \in \sigma$.*

  In questo caso, un guasto associato alla i-esima classe di guasto deve essersi verificato, poichè tutte le sequenze scattabili consistenti con l'osservazione contengono almeno un guasto in T_f^i. ■

Esempio 3.4.1. Consideriamo la RdP in Figura 2.5, precedentemente introdotta nell'Esempio 3.1.1. Sia $T_f = \{\varepsilon_{11}, \varepsilon_{12}\}$. Assumiamo che le due transizioni di guasto appartengano a due differenti classi di guasto, ossia $T_f^1 = \{\varepsilon_{11}\}$ e $T_f^2 = \{\varepsilon_{12}\}$.

Osservato $w = acd$, si ha $\Delta(w, T_f^1) = 0$ e $\Delta(w, T_f^2) = 2$, essendo $\hat{\mathcal{J}}(w) = \{(t_1t_5t_6, \varepsilon), (t_1t_5t_7, \varepsilon_{12}\varepsilon_{13})\}$ ed $\mathcal{S}(w) = \{t_1t_5t_6, t_1t_5\varepsilon_{12}\varepsilon_{13}t_7\}$.

Ora, si consideri $w = ab$. In questo caso $\Delta(w, T_f^1) = 1$ e $\Delta(w, T_f^2) = 0$, essendo $\hat{\mathcal{J}}(w) = \{(t_1t_2, \varepsilon)\}$ ed $\mathcal{S}(w) = \{t_1t_2, t_1t_2\varepsilon_8, t_1t_2\varepsilon_8\varepsilon_9, t_1t_2\varepsilon_8\varepsilon_9\varepsilon_{10}, t_1t_2\varepsilon_8\varepsilon_{11}\}$. ■

I seguenti due risultati, dimostrati in [7] per RdP non etichettate, valgono anche nel caso di RdP etichettate.

Proposizione 3.4.1. *Consideriamo una parola osservata $w \in L^*$.*

- $\Delta(w, T_f^i) \in \{0, 1\}$ *se e solo se per tutte le coppie $(M, y) \in \mathcal{M}(w)$ e per tutte le $t_f \in T_f^i$ vale che $y(t_f) = 0$.*

- $\Delta(w, T_f^i) = 2$ *se e solo se esiste una* $(M, y) \in \mathcal{M}(w)$ *e* $(M', y') \in \mathcal{M}(w)$ *tale che:*
  (i) esiste una $t_f \in T_f^i$ *tale che* $y(t_f) > 0$*,*
  (ii) per tutte le $t_f \in T_f^i$*,* $y'(t_f) = 0$*.*
- $\Delta(w, T_f^i) = 3$ *se e solo se per tutte le* $(M, y) \in \mathcal{M}(w)$ *esiste una* $t_f \in T_f^i$ *tale che* $y(t_f) > 0$*.* □

La seguente proposizione, sempre tratta da [7], mostrerà come fare distinzione tra gli stati 0 ed 1 del diagnosticatore.

Proposizione 3.4.2. *Per una RdP la cui sottorete non osservabile sia aciclica, sia* $w \in L^*$ *una parola osservata tale che per tutte le* $(M, y) \in \mathcal{M}(w)$ *si abbia* $y(t_f) = 0 \; \forall \, t_f \in T_f^i$*. Consideriamo l'insieme di vincoli*

$$\mathcal{T}(M) \;=\; \begin{cases} M + C_u \cdot z \geq \vec{0}, \\ \displaystyle\sum_{t_f \in T_f^i} z(t_f) > 0, \\ z \in \mathbb{N}^{n_u}. \end{cases} \tag{3.1}$$

- $\Delta(w, T_f^i) = 0$ *se* $\forall \; (M, y) \in \mathcal{M}(w)$ *l'insieme di vincoli* (3.1) *non ammette soluzione.*
- $\Delta(w, T_f^i) = 1$ *se* $\exists \; (M, y) \in \mathcal{M}(w)$ *tale che l'insieme dei vincoli* (3.1) *ammette soluzione.* □

Sulla base di questi risultati, se la sottorete non osservabile è aciclica, la diagnosi può essere effettuata semplicemente osservando l'insieme $\mathcal{M}(w)$ per qualunque parola w osservata e, nel caso in cui lo stato di diagnosi vale 0 o 1, valutando in aggiunta se il corrispondente insieme di vincoli interi (3.1) ammette una soluzione.

Esempio 3.4.2. Consideriamo la RdP in Figura 2.5 dove $T_f^1 = \{\varepsilon_{11}\}$ e $T_f^2 = \{\varepsilon_{12}\}$.

Sia $w = acd$. Si ha $\mathcal{M}(w) = \{(M_b, \vec{0}), (M_b, [0\ 0\ 0\ 0\ 1\ 1]^T)\}$, dove $M_b = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$ è stato determinato nell'Esempio 3.3.1. Si ha che $\Delta(w, T_f^1) = 0$, essendo verificato che $\mathcal{T}(M_b)$ non ammette soluzione.

Sia $w = ab$. In questo caso $\mathcal{M}(w) = \{(M_b, \vec{0})\}$, dove $M_b = [0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0]^T$ come nel caso precedente. Poichè $\mathcal{T}(M_b)$ ammette soluzione solo per la classe T_f^1, si ha che $\Delta(w, T_f^1) = 1$ e $\Delta(w, T_f^2) = 0$. ■

3.5 Grafo di raggiungibilità di base

In [7] è stato mostrato come, nel caso di RdP limitate, uno strumento utile per determinare la diagnosi online sia il *BRG*.

In questa sessione mostreremo come il BRG possa essere definito anche nel caso di RdP etichettate.

Il BRG è un grafo deterministico che ha tanti nodi quante sono il numero di marcature di base raggiungibili. A ciascun nodo è associata una distinta marcatura di base M ed un vettore riga con tanti elementi quante sono il numero di classi di guasto. Tali elementi del possono assumere solo valori binari: 1 se $\mathcal{T}(M)$ ammette soluzioni, 0 altrimenti.

Gli archi sono etichettati con eventi osservabili di $\mathcal{L}$ ed e-vettori. Più precisamente, esiste un arco da un nodo contenente una marcatura di base M ad un nodo contenente la marcatura di base M' se e solo se esiste una transizione t per la quale esiste una spiegazione a partire da M e lo scatto di t e di una delle sue spiegazioni minime porta ad M'. L'arco che porta da M ad M' è etichettato $(\mathcal{L}(t), e)$, dove $e \in Y_{\min}(M, t)$ ed $M' = M + C_u \cdot e + C(\cdot, t)$.

Si noti che il numero di nodi del BRG è sempre finito, poiché l'insieme delle marcature di base è un sottoinsieme dell'insieme di raggiungibilità, che è finito essendo la rete limitata. In aggiunta, il vettore riga di valori binari associati al nodo del BRG ci permette di fare distinzione tra lo stato di diagnosi 1 e 0.

I passi principali della determinazione del BRG nel caso di RdP etichettate sono riassunti nel seguente algoritmo.

Algoritmo 3.5.1. [Calcolo del BRG]

1. Etichetta il nodo iniziale (M_0, x_0) dove $\forall i = 1, \ldots, r$,

$$x_0(T_f^i) = \begin{cases} 1 & \text{se } \mathcal{T}(M_0) \text{ ammette soluzione,} \\ 0 & \text{altrimenti.} \end{cases}$$

Non assegnare alcun tag.

2. Finché esistono nodi senza tag,
seleziona un nodo senza tag e fai

2.1. sia M la marcatura del nodo (M, x),

2.2. per tutte le $l \in L$

2.2.1. per tutte le $t : \mathcal{L}(t) = l \wedge Y_{\min}(M, t) \neq \emptyset$, fai

- per tutti gli $e \in Y_{\min}(M, t)$, fai

- sia $M' = M + C_u \cdot e + C(\cdot, t)$,
- se $\nexists$ un nodo (M, x) con $M = M'$, fai
  - aggiungi un nuovo nodo al grafo contenente (M', x') dove $\forall i = 1, \ldots, r$,
    $$x'(T_f^i) = \begin{cases} 1 & \text{se } \mathcal{T}(M') \text{ ammette soluzione,} \\ 0 & \text{altrimenti.} \end{cases}$$
    e l'arco (l, e) da (M, x) a (M', x')
  - altrimenti
    - aggiungi un arco (l, e) da (M, x) a (M', x') se non esiste ancora

2.3. contrassegna il nodo con un tag.

3. Rimuovi tutti i tag. ■

L'algoritmo costruisce il BRG a partire dal nodo iniziale a cui corrisponde la marcatura iniziale ed un vettore binario, che definisce quali classi di guasto possono verificarsi in M_0. Ora, consideriamo tutte le etichette $l \in L$ tali che esista una transizione t con $\mathcal{L}(t) = l$ per cui esiste una spiegazione minima in M_0. Per ciascuna di queste transizioni calcoliamo la marcatura risultante dallo scatto di t da $M_0 + C_u \cdot e$, per ciascuna $e \in Y_{\min}(M_0, t)$. Se è ottenuta una coppia (marcatura, vettore binario) non contenuta nei nodi precedenti, viene aggiunto un nuovo nodo nel grafo. L'arco che va dal nodo iniziale al nuovo nodo è etichettato (l, e). La procedura è iterata sino a che tutte le marcature di base non sono considerate. Si noti che l'approccio descritto in questo capitolo richiede sempre di enumerare uno spazio di stato che è un sottoinsieme stretto dell'insieme di raggiungibilità. D'altronde, come usualmente negli approcci di diagnosi, non può essere evitata l'esplosione combinatoria.

Un esempio farà luce sui concetti introdotti precedentemente.

Esempio 3.5.1. Consideriamo la RdP in Figura 2.5, dove $T_o = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$, $T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}\}$, $T_f^1 = \{\varepsilon_{11}\}$ e $T_f^2 = \{\varepsilon_{12}\}$. La funzione di etichettatura è definita come segue: $\mathcal{L}(t_1) = a$, $\mathcal{L}(t_2) = \mathcal{L}(t_3) = b$, $\mathcal{L}(t_4) = \mathcal{L}(t_5) = c$, $\mathcal{L}(t_6) = \mathcal{L}(t_7) = d$.

Il BRG è mostrato in Figura 3.1. La notazione usata in questa figura è dettagliata nelle tabelle 3.1 e 3.2, indicanti rispettivamente le marcature di base ed i vettori di giustificazione. Ciascun nodo contiene una marcatura distinta ed un vettore riga bidimensionale, essendo due il numero delle classi di guasto. Ad esempio, il vettore binario $[0\ 0]$ è associato ad M_0 poiché $\mathcal{T}(M_0)$ non ammette soluzione per entrambe le classi di guasto. Dal nodo M_0 al nodo M_1 c'è un arco etichettato a

M_0	$[\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ]^T$
M_1	$[\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ ]^T$
M_2	$[\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ ]^T$
M_3	$[\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ ]^T$
M_4	$[\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ ]^T$
M_5	$[\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ ]^T$
M_6	$[\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ ]^T$

Tabella 3.1: Le marcature del BRG in Figura 3.1.

	ε_8	ε_9	ε_{10}	ε_{11}	ε_{12}	ε_{13}
e_1	0	0	0	0	1	1
e_2	1	1	1	0	0	0
e_3	1	0	0	1	0	0

Tabella 3.2: Gli e-vettori del BRG in Figura 3.1.

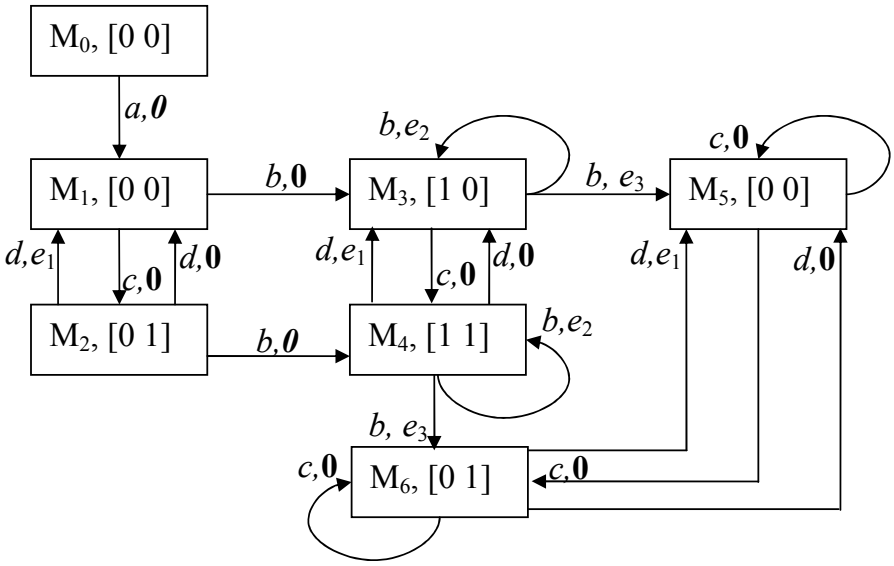

Figura 3.1: Il BRG della RdP in Figura 2.5.

e con il vettore nullo come spiegazione minima. Il nodo contenente la marcatura di base M_2 ha un vettore binario $[0\ 1]$, poiché $\mathcal{T}(M_2)$ ammette soluzione solo per T_f^2. Il nodo $(M_2, [0\ 1])$ ha due archi uscenti entrambi etichettati con d ed entrambi diretti verso $(M_1, [0\ 0])$ con due differenti spiegazioni minime, $\vec{0}$ ed e_1, rispettivamente. Infine, un ulteriore altro arco uscente $(b, \vec{0})$ è diretto al nodo $(M_4, [1\ 1])$. ■

Il BRG può essere usato per effettuare una diagnosi online, come mostrato nell'esempio seguente. L'algoritmo diagnostico viene riportato nel seguito per completezza.

Algoritmo 3.5.2. ***[****Diagnosi mediante il BRG****]***

1. *Sia* $w = \varepsilon$.
2. *Sia* $\mathcal{M}(w) = \{(M_0, \vec{0})\}$.
3. *Attendi sino allo scatto di una nuova transizione osservabile. Sia* l *l'evento osservato.*
4. *Sia* $w' = w$ *e* $w = w'l$.
5. *Sia* $\mathcal{M}(w) = \emptyset$, ***[****Determinazione dell'insieme* $\mathcal{M}(w)$***]***
6. *Per tutti i nodi contenenti* M' *:* $(M', y') \in \mathcal{M}(w')$*, fai*
  6.1. *per tutti gli archi uscenti dal nodo con* M'*, fai*
    6.1.1. *sia* M *la marcatura del nodo raggiunto ed* e *l'e-vettore minimo dell'arco incidente da* M' *ad* M*,*
    6.1.2. *Per tutte le* y' *tali che* $(M', y') \in \mathcal{M}(w')$*, fai*
      6.1.2.1. *sia* $y = y' + e$*,*
      6.1.2.2. *sia* $\mathcal{M}(w) = \mathcal{M}(w) \cup \{(M, y)\}$*,*
7. *per tutte le* $i = 1, \ldots, r$*, fai* ***[****Determinazione dello stato di diagnosi****]***
  7.1. *se* $\forall\ (M, y) \in \mathcal{M}(w) \wedge \forall t_f \in T_f^i$ *vale* $y(t_f) = 0$*, fai*
    7.1.1. *se* $\forall\ (M, y) \in \mathcal{M}(w)$ *vale che* $x(i) = 0$*, dove* x *è il vettore binario nel nodo* M*, fai*
      7.1.1.1. *sia* $\Delta(w, T_f^i) = 0$*,*
    7.1.2. *altrimenti*
      7.1.2.1. *sia* $\Delta(w, T_f^i) = 1$*,*
  7.2. *se* $\exists\ (M, y) \in \mathcal{M}(w)$ *e* $(M', y') \in \mathcal{M}(w)$ *vincolato a:*
    (i) $\exists t_f \in T_f^i$ *tale che* $y(t_f) > 0$*,*
    (ii) $\forall t_f \in T_f^i$*,* $y'(t_f) = 0$*, fai*
    7.2.1. *sia* $\Delta(w, T_f^i) = 2$*,*
  7.3. *se* $\forall\ (M, y) \in \mathcal{M}(w)\ \exists t_f \in T_f^i$ *:* $y(t_f) > 0$*, fai*
    7.3.1. *sia* $\Delta(w, T_f^i) = 3$*.*
8. *Ritorna al passo 3.* ■

Esempio 3.5.2. Si consideri la RdP in Figura 2.5 ed il suo BRG in Figura 3.1. Sia $w = \varepsilon$. Osservando il BRG si può stabilire che $\Delta(\varepsilon, T_f^1) = \Delta(\varepsilon, T_f^2) = 0$, essendo entrambi gli argomenti del vettore riga associato ad M_0 pari a 0.

Ora, si consideri $w = ab$. In tal caso $\mathcal{M}(w) = \{(M_3, \vec{0})\}$. Si ha che $\Delta(ab, T_f^1) = 1$ e $\Delta(ab, T_f^2) = 0$, essendo il vettore riga associato al nodo pari a $[1\ 0]$.

Per ultimo, si consideri l'osservazione della parola $w = abbc$. In questo caso si ha $\Delta(abbc, T_f^1) = 2$ e $\Delta(abbc, T_f^2) = 1$. Questo perché $\mathcal{M}(w) = \{(M_4, y_1), (M_5, y_2), (M_6, y_3)\}$, dove $y_1 = e_2$, $y_2 = y_3 = e_3$. I vettori riga associati ad M_4, M_5 ed M_6 sono rispettivamente $[1\ 1]$, $[0\ 0]$ ed $[0\ 1]$. ■

Capitolo 4

Diagnosticabilità mediante reti di Petri

Sommario

In questo capitolo tratteremo il problema della diagnosticabilità di reti di Petri (RdP) limitate. Verranno date condizioni necessarie e sufficienti per la determinazione della proprietà di diagnosticabilità e, in seguito, presentato un metodo basato sull'analisi del *Grafo di Raggiungibilità di Base Modificato* (MBRG) e del *Diagnosticatore di Raggiungibilità di Base* (BRD).

4.1 Inquadramento del problema

Si assuma, come nel capitolo precedente, che l'insieme delle transizioni sia suddiviso tra transizioni osservabili e non osservabili, ossia $T = T_o \cup T_u$. In aggiunta, si assuma che l'insieme delle transizioni non osservabili sia suddiviso ulteriormente in due sottoinsiemi $T_u = T_f \cup T_{reg}$, indicanti rispettivamente l'insieme delle transizioni modellanti fenomeni di guasto e l'insieme delle transizioni regolari. Quest'ultimo insieme include tutte quelle transizioni silenziose ma corrispondenti a dinamiche regolari. Il sottoinsieme T_f è a sua volta suddiviso in r distinti sottoinsiemi T_f^i, dove $i = 1, \ldots, r$, che caratterizzano le differenti classi di guasto.

Definizione 4.1.1. *Un sistema di RdP $\langle N, M_0 \rangle$ è detto non bloccante dopo un guasto, se per ogni marcatura morta $M \in R(N, M_0)$ $\nexists \sigma \mid M_0[\sigma\rangle M \wedge \nexists t_f \in$*

$T_f^i \mid t_f \in \sigma$. ■

Se esiste una marcatura morta, dunque non è possibile raggiungerla tramite una sequenza di transizioni contenente un guasto.

Definizione 4.1.2. *Una RdP non bloccante dopo un guasto* $\langle N, M_0 \rangle$ *è detta* diagnosticabile in riferimento alla classe di guasto T_f^i *se non esistono due sequenze* σ_1 *and* σ_2 *in* T^* *soddisfacenti le seguenti condizioni:*

i) $\mathcal{L}(\sigma_1) = \mathcal{L}(\sigma_2)$,

ii) $\forall t_f \in T_f^i$, $t_f \notin \sigma_1$,

iii) $\exists$ *almeno una* $t_f \in T_f^i$ *tale che* $t_f \in \sigma_2$,

iv) σ_2 *può essere arbitrariamente lunga dopo un guasto* $t_f \in T_f^i$. ■

Definizione 4.1.3. *Un sistema di RdP* $\langle N, M_0 \rangle$ *è detto* diagnosticabile *se è diagnosticabile in riferimento a tutte le classi di guasto.* ■

Si noti che la diagnosticabilità di un sistema non implica l'esser capaci di fare distinzione tra due transizioni della stessa classe di guasto. Bensì, essa implica semplicemente che, se una o più transizioni in una data classe di guasto sono scattate, allora, dopo un finito numero di osservazioni, siamo capaci di stabilire che almeno una transizione di tale classe di guasto è scattata. In questo capitolo, ci occuperemo del problema di fornire condizioni necessarie e sufficienti per la diagnosticabilità. In particolare, considereremo RdP etichettate che soddisfano le condizioni seguenti.

A1) Il sistema di rete $\langle N, M_0 \rangle$ è *limitato* e *non termina in stallo in seguito allo scatto di nessuna transizione*.

A2) La $rete\ T_u\text{-}indotta$ è *aciclica*.

A3) La funzione di etichettatura $\mathcal{L} : T_o \rightarrow L$ può associare la stessa etichetta a differenti transizioni.

A4) La struttura di N è nota come la sua marcatura iniziale M_0.

4.2 Grafo di Raggiungibilità di Base Modificato

In [7] è stato mostrato come, nel caso di RdP limitate, un utile strumento per effettuare la diagnosi sia il *Grafo di Raggiungibilità di Base* (BRG). In particolare, si è visto come esso ci permetta di effettuare la gran parte della procedura offline. Nel seguito, mostreremo come tale grafo non sia sufficiente per effettuare concretamente l'analisi di diagnosticabilità della rete.

Per questa motivazione, il BRG necessita di essere modificato, se si vuole utilizzarlo come strumento ausiliario per stabilire se il sistema in analisi sia diagnosticabile. Definiamo,così, un nuovo grafo, detto *Grafo di Raggiungibilità Modificato* (MBRG).

L'MBRG è un grafo deterministico i cui nodi contengono due elementi (M, x): $M \in \mathbb{N}^m$ è la marcatura definita come segue ed x è un vettore riga di $\{0,1\}^{|T_f|}$ dove $x(i) = 1$ se $\mathcal{T}(M)$ in (3.1) è ammissibile in riferimento alla i-esima classe, $x(i) = 0$ altrimenti.

La marcatura M del nodo è calcolata, analogamente alla costruzione del BRG, come marcatura di base, assumendo che tutte le transizioni di guasto siano osservabili. Questo significa che le spiegazioni minime sono limitate alle sole transizioni in T_{reg}.

Nel seguito indicheremo con $Y_{\min}^{mod}(M,t)$ l'insieme degli e-vettori minimi limitati a T_{reg} e C_{reg} la restrizione della matrice di incidenza a T_{reg}.

Gli archi possono essere etichettati in due differenti modi, in maniera dipendente dall'evento in analisi.

Nel caso di eventi corrispondenti allo scatto di transizioni in T_o, l'etichetta contiene tre informazioni riassunte con $(l(t), e)$, dove $l \in L$ è l'etichetta osservata, t è la transizione etichettata l il cui scatto al nodo di ingresso è abilitato dalla sequenza di transizioni regolari con vettore di scatto $e \in Y_{\min}^{mod}(M,t)$ e che porta alla marcatura del nodo di uscita.

Nel caso di eventi corrispondenti a transizioni di guasto, l'etichetta contiene solo due informazioni riassunte con (t_f, e), dove $t_f \in T_f$ è la transizione di guasto il cui scatto al nodo di ingresso è abilitato da una sequenza con vettore di scatto $e \in Y_{\min}^{mod}(M,t)$ e che porta alla marcatura del nodo di arrivo.

Un algoritmo formale per la costruzione dell'MBRG è il seguente.

Algoritmo 4.2.1. [Calcolo dell'MBRG]

1. Etichetta il nodo iniziale (M_0, x_0) dove $\forall i = 1, \ldots, r$,

$$x_0(T_f^i) = \begin{cases} 1 & \text{se } \mathcal{T}(M_0) \text{ ammette soluzione,} \\ 0 & \text{altrimenti.} \end{cases}$$

Non assegnare alcun tag.

2. Sinché esiste un nodo senza tag

2.1. seleziona un nodo senza tag,

2.2. sia (M, x) il nodo selezionato,

2.3. per tutte le $l \in L$

2.3.1. per tutte le $t : L(t) = l \wedge Y_{\min}^{mod}(M, t) \neq \emptyset$, fai

- per tutte le $e \in Y_{\min}^{mod}(M, t)$, fai
  - sia $M' = M + C_{reg} \cdot e + C(\cdot, t)$,
  - se $\nexists$ un nodo con M', fai
    - aggiungi un nuovo nodo al grafo contenente la coppia (M', x'), dove $\forall i = 1, \ldots, r$

$$x'(T_f^i) = \begin{cases} 1 & \text{se } \mathcal{T}(M') \text{ ammette soluzione,} \\ 0 & \text{altrimenti.} \end{cases}$$

  - aggiungi un arco $(l(t), e)$ dal nodo (M, x) al nodo (M', x')

2.4. per $i = 1, \ldots, r \ : \ x(T_f^i) = 1$

2.4.1. per tutte le $t_f \in T_f^i : Y_{\min}^{mod}(M, t_f) \neq \emptyset$, fai

- per tutte le $e \in Y_{\min}^{mod}(M, t_f)$, fai
  - sia $M' = M + C_{reg} \cdot e + C(\cdot, t_f)$,
  - se $\nexists$ un nodo senza M', fai
    - aggiungi un nuovo nodo al grafo contenente la coppia (M', x'), dove $\forall i = 1, \ldots, r$

$$x'(T_f^i) = \begin{cases} 1 & \text{se } \mathcal{T}(M') \text{ ammette soluzione,} \\ 0 & \text{altrimenti.} \end{cases}$$

  - aggiungi un arco (t_f, e) dal nodo (M, x) al nodo (M', x')

2.5. contrassegna con un tag il nodo (M, x) .

3. Rimuovi tutti i tag. ■

L'algoritmo costruisce l'MBRG a partire da un nodo iniziale a cui corrisponde la marcatura iniziale ed al contempo un vettore binario che definisce quale classe di guasto possa scattare in M_0. Ora, si considerino tutte le etichette $l \in L$ (passo 2.3) e tutte le classi di guasto $i = 1, \ldots, r$ (passo 2.4), tali che esista una transizione t con $L(t) = l$ o una transizione di guasto $t_f \in T_f^i$ per cui esista una spiegazione minima in M_0. Per ciascuna di queste transizioni, che possono essere $t \in T_o$ o

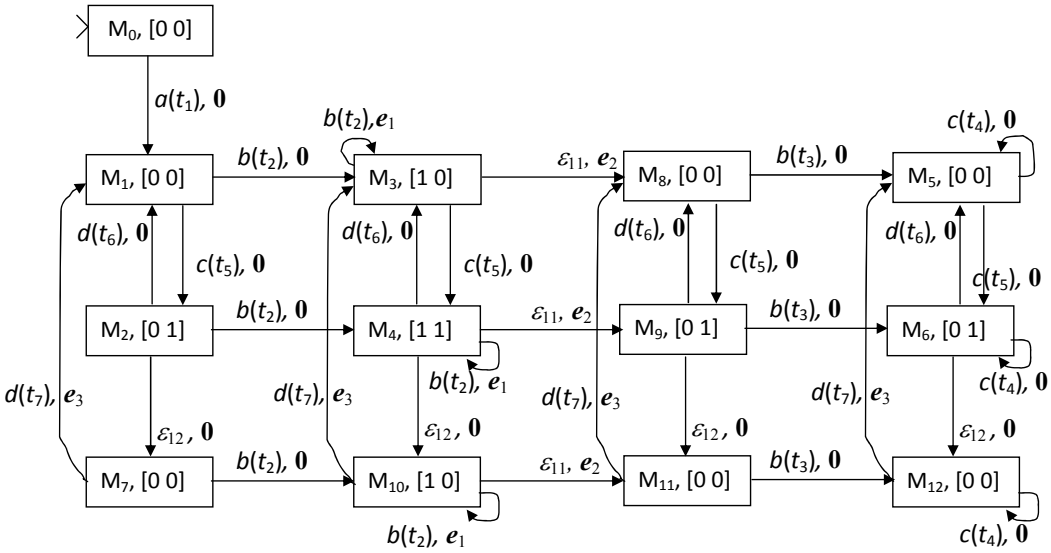

Figura 4.1: L'MBRG della RdP in Figura 2.5.

M_0	$[\ 2\ 0\ 0\ 0\ 0\ 0]^T$
M_1	$[\ 1\ 0\ 1\ 0\ 0\ 0]^T$
M_2	$[\ 0\ 0\ 2\ 0\ 0\ 0]^T$
M_3	$[\ 1\ 0\ 0\ 0\ 1\ 0]^T$
M_4	$[\ 0\ 0\ 1\ 0\ 1\ 0]^T$
M_5	$[\ 0\ 0\ 0\ 0\ 2\ 0]^T$

Tabella 4.1: Le marcature di base dell'MBRG in Figura 4.1.

$t_f \in T_f^i$, determiniamo la marcatura M' dal suo scatto a partire da $M_0 + C_u \cdot e$ ($e \in Y_{\min}^{mod}(M_0, t)$ o $e \in Y_{\min}^{mod}(M_0, t_f)$, rispettivamente). Se è raggiunta una nuova coppia (marcatura, vettore binario), viene aggiunto un nuovo nodo al grafo, contenente la marcatura risultate M' ed il vettore x' corrispondente. L'arco che va dal nodo iniziale al nuovo nodo viene etichettato $(\mathcal{L}(t), e)$ o (t_f, e), in dipendenza dell'evento considerato. La procedura è iterata sinché tutti i nodi non vengono esaminati. Si noti che, se la rete è limitata, la procedura termina in un numero finito di passi, poiché il numero di nodi è limitato superiormente dalla cardinalità dell'insieme di raggiungibilità $R(N, M_0)$.

Esempio 4.2.1. In Figura 4.1 è mostrato l'MBRG corrispondente alla RdP in Figura 2.5, introdotta nell'Esempio 3.1.1. La notazione usata in Figura 4.1 è dettagliata nelle tabelle 4.1 e 4.2.

	ε_4	ε_6	ε_8
e_1	1	0	0
e_2	0	1	0
e_3	0	0	1

Tabella 4.2: Gli e-vettori minimi modificati dell'MBRG in Figura 4.1.

Ciascun nodo contiene una marcatura differente ed uno scalare (poiché c'è una sola classe di guasto). Come nell'esempio, lo scalare 1 è associato ad M_0 poiché $\mathcal{T}(M_0)$ ammette soluzione.

Gli archi sono etichettati o dalla coppia (etichetta (transizione corrispondente), e-vettore minimo modificato corrispondente), si veda per esempio $(a(t_1), e_1)$ dal nodo iniziale, o dalla coppia (transizione non osservabile, e-vettore minimo modificato corrispondente), si veda per esempio (ε_7, e_2) da M_1.

Infine, si osservi che non tutte le marcature dei nodi sono marcature di base. Precisamente, $M_0, M_1,$ ed M_2 sono marcature di base, mentre M_3, M_4 ed M_5 sono marcature raggiunte da marcature di base tramite lo scatto delle transizioni di guasto ε_5 e ε_7. Ciò mostra che la memoria richiesta, necessaria per la risoluzione del problema di diagnosticabilità, è superiore a quella necessaria per la risoluzione del problema di diagnosi. Si noti che il numero di marcature nell'MBRG sarebbe comunque pari al numero di marcature consistenti solo nel caso peggiore, ma in generale è inferiore, come nell'esempio trattato. ■

4.3 Diagnosticatore di Raggiungibilità di Base

In questo capitolo definiamo il diagnosticatore di cui faremo uso in seguito.

Definizione 4.3.1. *Il Diagnosticatore di Raggiungibilità di Base (BRD) è un grafo deterministico, dove ciascun nodo contiene le informazioni seguenti:*

i) una o più triple (M, x, h)*, dove:*

- M *è una marcatura di base;*
- $x \in \{0,1\}^{|T_f|}$ *è un vettore riga il cui* i*-esimo elemento è pari a* 1*, se* $\mathcal{T}(M)$ *ammette soluzione in riferimento alla* i*-esima classe di guasto, o è pari a* 0 *altrimenti;*

– *$h \in \{N, F\}^{|T_f|}$ è un vettore riga il cui i-esimo elemento è pari a N, se raggiungendo M da M_0 non si è verificato nessun guasto di T_f^i, o è pari ad F altrimenti;*

ii) *r etichette Δ_i, per $i = 1, \ldots, r$, che rappresentano lo stato di diagnosi del nodo in riferimento alle r classi di guasto.*

Infine, gli archi sono etichettati con un simbolo di $\mathcal{L}$. ■

È un grafo deterministico che, se utilizzato in aggiunta all'MBRG, ci permette di formulare condizioni necessarie e sufficienti per la diagnosticabilità.

Il BRD può essere facilmente determinato a partire dall'MBRG. In particolare, i valori di M ed x sono leggibili dall'MBRG, osservando semplicemente i nodi contenenti le marcature di base.

Il valore di h può essere dedotto, osservando il percorso (o i differenti percorsi nel caso ce ne fossero più di uno) da M_0 al corrispondente valore di M (indicato con $M_0 \rightsquigarrow M$). Se esiste un percorso $M_0 \rightsquigarrow M$ contenente transizioni di guasto della classe i-esima, allora alla coppia (M, x) è associato un valore $h(i) = F$. Se, viceversa, esiste un percorso $M_0 \rightsquigarrow M$ non contenente alcuna transizione di guasto della classe i-esima, allora alla coppia (M, x) è associato un valore $h(i) = N$. Si noti che, poiché in generale possono esistere più di un percorso da M_0 ad M, uno contenente un guasto di T_f^i ed uno no, la coppia (M, x) può essere presente due volte nello stesso nodo, ma con $h(i) = F$ ed $h(i) = N$.

Si noti che, durante la costruzione del BRD, vengono considerate solo le marcature di base e non tutte le marcature dell'MBRG. Di conseguenza, per ciascuna etichetta $l \in \mathcal{L}$ dobbiamo considerare solo le marcature raggiunte tramite lo scatto di $Y_{min}(M, t)$ per tutte le transizioni t tali che $l = \mathcal{L}(t)$.

Lo stato di diagnosi per ciascuna classe di guasto è banalmente ottenuto per definizione, semplicemente osservando i due ultimi elementi di tutte le triple del nodo.

Il seguente algoritmo sintetizza i passi principali per la costruzione del BRD. Si noti che, al fine di semplificare la notazione, si è assunto che ciascuna classe di guasto includa un'unica transizione, per cui $|T_f| = r$. L'estensione al caso più generale è banale e non viene riportata per motivi di brevità.

Algoritmo 4.3.1. [Determinazione del BRD]

1. Etichetta il nodo iniziale $d_0 = (M_0, x_0, h_0)$, $h_0 = N^r$.
Per $i = 1, \ldots, r$, se $x_0(i) = 0$ allora $\Delta_i = 0$, altrimenti $\Delta_i = 1$.
Non assegnargli alcun tag.
2. finché esistono nodi senza tag
2.1. seleziona un nodo d senza tag e fai
2.2. per tutte le $l \in L$
2.2.1. per tutte le $M \in d$: $Y_{\min}(M, t) \neq \emptyset$ e per le transizioni $t : L(t) = l$
• per tutte le triple con marcatura M in d
• sia $\tilde{d} = \emptyset$
• per tutti gli archi uscenti (M, x) nell'MBRG etichettati l, fai
• sia (M', x') il nodo raggiunto nell'MBRG,
• sia

$$\begin{cases} h'(i) = N & \text{se } h(i) = N \\ h'(i) = F & \text{se } h(i) = F \end{cases}$$

• sia $\tilde{d} = \tilde{d} \cup \{(M', x', h')\}$
• per tutti i percorsi uscenti da (M, x) nell'MBRG etichettati $\sigma_f l$ tali che $\pi(\sigma_f) \in Y_{\min}(M, t)$ e $L(t) = l$,
• sia (M', x') il nodo finale nell'MBRG,
• sia

$$\begin{cases} h'(i) = N & \text{se } h(i) = N \wedge t_{f_i} \notin M \rightsquigarrow M' \\ h'(i) = F & \text{se } h(i) = F \\ h'(i) = F & \text{se } h(i) = N \wedge t_{f_i} \in M \rightsquigarrow M' \end{cases}$$

• sia $\tilde{d} = \tilde{d} \cup \{(M', x', h')\}$
• se $\forall\ M' \in \tilde{d}$ vale $h'(i) = N$ e $x'(i) = 0$, allora
• sia $\Delta_i = 0$
• altrimenti, se $\forall M' \in \tilde{d}$ vale $h'(i) = N$ ed $x'(i) = 1$, allora
• sia $\Delta_i = 1$
• altrimenti, se $\exists (M', x', h') \in \tilde{d} : h'(i) = N$ ed $\exists (M'', x'', h'') \in \tilde{d} : h''(i) = F$, allora
• sia $\Delta_i = 2$
• altrimenti, se $\forall\ M' \in \tilde{d}$ vale $h'(i) = F$, allora
• sia $\Delta_i = 3$
2.2.2 se $\nexists$ un nodo $\bar{d} = \tilde{d}$ nel grafo, allora
• aggiungi un nuovo nodo $\tilde{d}$ al grafo
2.2.3 aggiungi un arco l da d a $\tilde{d}$
2.3 contrassegna il nodo d con un tag.
2.4 Vai al passo 2.1.
3. Rimuovi tutti i tag. ■

L'algoritmo costruisce il BRD a partire dal nodo iniziale, a cui corrisponde la tripla (M_0, x_0, h_0), dove M_0 ed x_0 corrispondono al nodo iniziale dell'MBRG, ed il terzo elemento è inizialmente posto pari a $h_0 = N^r$. Il suo stato di diagnosi Δ_i è posto a zero se non esiste una sequenza di transizioni osservabili che abilitano una transizione di guasto in T_f^i dalla marcatura iniziale, ossia se l'elemento di x_0 associato all'unica (per assunzione) transizione di guasto $t_{f_i} \in T_f^i$ è nullo, altrimenti Δ_i è posto ad uno.

A partire dal nodo iniziale ed osservando l'MBRG, prestiamo attenzione all'insieme delle marcature di base raggiungibili in seguito allo scatto di transizioni con etichetta l in M_0, immediatamente o dopo lo scatto di una o più transizioni di guasto.

Il nuovo nodo sarà composto da tutte le triple (M', x', h') tali che la coppia (M', x') sia raggiunta nell'MBRG con lo scatto di transizioni etichettate l in M_0 o con lo scatto di una spiegazione minima, contenente una o più transizioni di guasto, e, successivamente, l'etichetta l considerata; h' è definito considerando h_0 e tutti i percorsi $M_0 \rightsquigarrow M'$ nell'MBRG.

Per ultimo, lo stato di diagnosi Δ_i dipende per ciascun nodo dall'elemento i-esimo dei due vettori x ed h di tutte le marcature appartenenti al nodo.

La procedura viene iterata sino a che non vengono esplorati tutti i nodi.

Esempio 4.3.1. In Figura 4.2 è riportato il BRD della RdP in Figura 2.5, dove $T_{f1} = \{\varepsilon_1 1\}$ e $T_{f2} = \{\varepsilon_1 2\}$.

Il nodo iniziale contiene la tripla

$$(M_0, [0, 0], [N, N])$$

ed il suo stato di diagnosi è $\Delta = 0$, essendo x_0 identicamente nullo. Da questo nodo è abilitata la transizione a e porta al nodo

$$(M_0, [0, 0], [N, N]),$$

con medesimo valore dello stato di diagnosi. Da quest'ultimo sono abilitate b e c e portano rispettivamente ai nodi

$$(M_3, [1, 0], [N, N])\ e\ (M_2, [0, 1], [N,\ N]).$$

Lo stato di diagnosi di questi due nodi è $\Delta = [1, 0]$ e $\Delta = [0, 1]$, infatti il sono raggiunti senza che nessuna transizione di guasto scatti, ma per essi vale, rispettivamente, $x_3(1) = 1$ ed $x_2(2) = 1$. A partire dal primo di quest'ultima coppia di

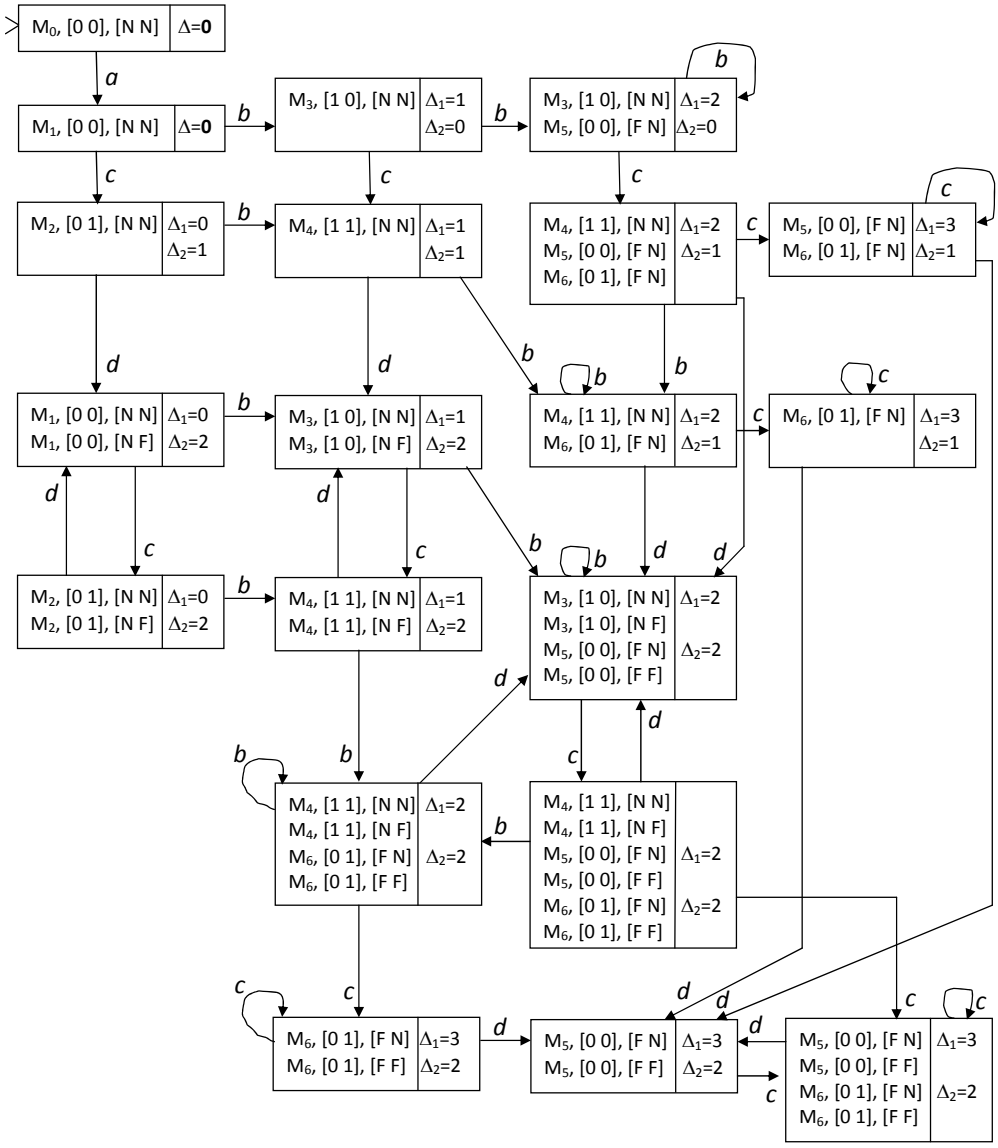

Figura 4.2: Il BRD della rete in Figura 2.5.

nodi, lo scatto di b porta al nodo

$$[(M_3, [1,0], [N,N]),\ (M_5, [0,0], [F,N])],$$

con stato di diagnosi $\Delta = [2,0]$ poiché esso è composto da due triple, la prima con $h_3(1) = N$ e la seconda con $h_5(1) = F$.

Infine, si noti come lo scatto della parola cc porti al nodo

$$[(M_5, [0,0], [F,N]),\ (M_6, [0,1], [F,N])],$$

il cui stato di diagnosi vale $\Delta = [3,1]$, poiché per le due triple appartenenti al nodo si ha $h_5(1) = h_6(1) = F$. ■

4.4 Condizioni necessarie e sufficienti per la diagnosticabilità

In questa sezione, forniamo condizioni necessarie e sufficienti per la diagnosticabilità e mostriamo come tali condizioni possano essere verificate tramite l'utilizzo del BRD congiuntamente all'MBRG. Queste condizioni sono basate sulla nozione di ciclo indeterminato, che formalizziamo come segue.

Definizione 4.4.1. *Sia γ un ciclo nel BRD con proiezione osservabile $\rho \in L^*$ e sia $p \in L^*$ un percorso dal nodo iniziale ad uno appartenente al suddetto ciclo. Il ciclo γ è* incerto *in riferimento alla classe di guasto T_f^i solo se include stati con $\Delta_i = 2$, o $\Delta_i = 1$, o $\Delta_i = 1$ e $\Delta_i = 2$.* ■

Definizione 4.4.2. *Sia γ un ciclo incerto nel BRD con proiezione osservabile $\rho \in L^*$ e sia $p \in L^*$ un percorso dal nodo iniziale ad uno appartenente al suddetto ciclo. Se nell'MBRG esistono due cicli γ_1 e γ_2 soddisfacenti le tre seguenti condizioni:*

(i) la loro proiezione osservabile è pari a ρ;

(ii) esistono due percorsi p_1 e p_2 con proiezione osservabile p, che dal nodo iniziale dell'MBRG abilitino γ_1 e γ_2;

(iii) Entrambi γ_2 e p_2 non contengono un guasto in T_f^i, mentre o γ_1 o p_1 o entrambi contengono un guasto in T_f^i. ■

Esempio 4.4.1. In Figura 4.3 e 4.4 sono riportati i cicli incerti rispettivamente per la prima e la seconda classe di guasto. Si considerino, inizialmente, due cicli incerti per la prima classe di guasto. Dunque, si osservi

$$\gamma = [(M_3, [1\,0], [N\ N]), (M_5, [0\,0], [F\ N])] \xrightarrow{b} [(M_3, [1\,0], [N\ N]), (M_5, [0\,0], [F\ N])],$$

per cui

$$\rho = b \text{ e } p = (M_0, [0\,0], [N\ N]) \xrightarrow{a} (M_1, [0\,0], [N\ N]) \xrightarrow{b} (M_3, [1\,0], [N\ N]) \xrightarrow{b}.$$

Osservando l'MBRG in Figura 4.1, è facile verificare che le condizioni della Definizione 4.4.2 non sono soddisfatte. Questo poiché non esiste un percorso p_1, contenente il guasto ε_{11}, avente la stessa proiezione osservabile di p e che, in aggiunta, abiliti γ_1 tale che $P_o(\gamma_1) = \rho$. Di conseguenza il ciclo non è indeterminato.

Si consideri il ciclo

$$[(M_1, [0\,0], [N\ N]), (M_1, [0\,0], [N\ F])] \xrightarrow{c} [(M_2, [0\,1], [N\ N]), (M_2, [0\,1], [N\ F])] \xrightarrow{d} [(M_1, [0\,0], [N\ N]), (M_1, [0\,0], [N\ F])],$$

per il quale

$$\rho = cd \text{ e } p = (M_0, [0\,0], [N\ N]) \xrightarrow{a} (M_1, [0\,0], [N\ N]) \xrightarrow{c} (M_2, [0\,1], [N\ N]) \xrightarrow{d}.$$

In questo caso, le tre condizioni della Definizione 4.4.2 sono soddisfatte e dunque il ciclo ρ è indeterminato. Infatti, l'MBRG mostra due cicli

$$\gamma_1 = (M_1, [0\,0]) \overset{c(t_5)}{\rightarrow} (M_2, [0\,1]) \overset{\varepsilon_{12}}{\rightarrow} (M_7, [0\,0]) \overset{d(t_7)}{\rightarrow} (M_1, [0\,0])$$

$$\gamma_2 = (M_1, [0\,0]) \overset{c(t_5)}{\rightarrow} (M_2, [0\,1]) \overset{d(t_6)}{\rightarrow} (M_1, [0\,0])$$

aventi la stessa proiezione osservabile ρ ed esistono due percorsi

$$p_1 = p_2 = (M_0, [0\,0]) \overset{a(t_1)}{\rightarrow} (M_1, [0\,0]) \overset{c(t_5)}{\rightarrow} (M_2, [0\,1]) \overset{d(t_6)}{\rightarrow}$$

aventi la stessa proiezione osservabile di p e tali da abilitare γ_1 e γ_2 dal nodo iniziale. In aggiunta p_2 e γ_2 non contengono la transizione di guasto ε_{11}, mentre γ_1 si.

Si consideri, ora, un ciclo incerto per la seconda classe. Sia

$$\gamma = (M_6, [0\ 1], [F\ N]) \xrightarrow{c} (M_6, [0\ 1], [F\ N])$$

il ciclo indeterminato nel BRD, per il quale

$\rho = c$ e $p = (M_0, [0\ 0], [N\ N]) \xrightarrow{a} (M_1, [0\ 0], [N\ N]) \xrightarrow{b} (M_3, [1\ 0], [N\ N]) \xrightarrow{c}$ $(M_4, [1\ 1], [N\ N]) \xrightarrow{b} [(M_4, [1\ 1], [N\ N]), (M_6, [0\ 1], [F\ N])] \xrightarrow{c}$.

Osservando l'MBRG in Figura 4.1, si deduce che tale ciclo è indeterminato dal momento che le condizioni richieste dalla Definizione 4.4.2 sono soddisfatte. Nell'MBRG, in effetti, esistono due cicli

$$\gamma_1 = (M_6, [0\ 1]) \overset{c(t_4)}{\rightarrow} (M_6, [0\ 1]) e\ \ \gamma_2 = (M_{12}, [0\ 0]) \overset{c(t_4)}{\rightarrow} (M_{12}, [0\ 0])$$

aventi la stessa proiezione osservabile ρ ed esistono due percorsi

$$p_1 = (M_0, [0\,0]) \overset{a(t_1)}{\rightarrow} (M_1, [0\,0]) \overset{b(t_2)}{\rightarrow} (M_3, [1\,0]) \overset{c(t_5)}{\rightarrow} (M_4, [1\,1]) \overset{\varepsilon_{11}}{\rightarrow} (M_9, [0\,1]) \overset{b(t_3)}{\rightarrow}$$

e $p_2 = (M_0, [0\ 0]) \underrightarrow{a(t_1)} (M_1, [0\ 0]) \overset{b(t_2)}{\rightarrow} (M_3, [1\ 0]) \overset{c(t_5)}{\rightarrow} (M_4, [1\ 1]) \overset{\varepsilon_{11}}{\rightarrow} (M_9, [0\ 1])$ $\overset{b(t_3)}{\rightarrow} (M_6, [0\ 1]) \overset{\varepsilon_{12}}{\rightarrow}$,

aventi la stessa proiezione osservabile di p e tali da abilitare a partire dal nodo iniziale γ_1 e γ_2. In aggiunta, p_2 e γ_2 non contengono la transizione di guasto ε_{12}, mentre p_1 si. ■

Teorema 4.4.1 ([8])**.** *Un sistema di rete $\langle N, M_0 \rangle$, soddisfacente le assunzioni da (A1) ad (A4), è* diagnosticabile *in riferimento alla classe T_f^i se e solo se il suo BRD non ha cicli indeterminati in riferimento a T_f^i.* ■

Corollario 4.4.1 ([8])**.** *Un sistema di rete $\langle N, M_0 \rangle$ soddisfacente le assunzioni da (A1) ad (A4) è* diagnosticabile *se e solo se il suo BRD non ha cicli che sono indeterminati rispetto ad ogni classe di guasto.* ■

Poiché in un grafo un ciclo è sempre associato ad una ed una sola componente fortemente connessa, l'idea iniziale per la realizzazione di un algoritmo di ricerca dei cicli è stata quella di esplorare tutti i possibili percorsi appartenenti in tali singole componenti. Per tale scopo, si è analizzato l'algoritmo Tarjan ([23]), appunto

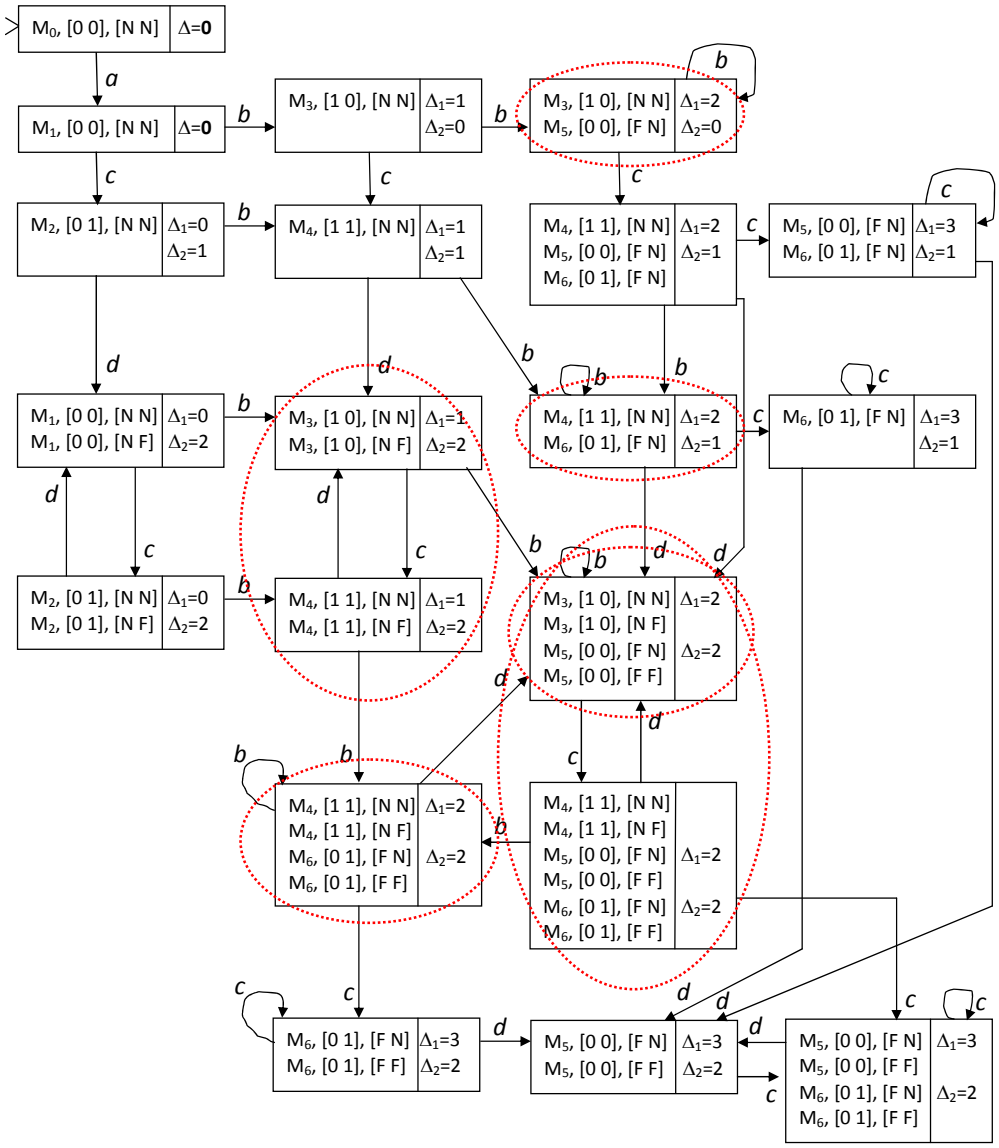

Figura 4.3: BRD della RdP in Figura 2.5 con i cicli incerti per la prima classe di guasto.

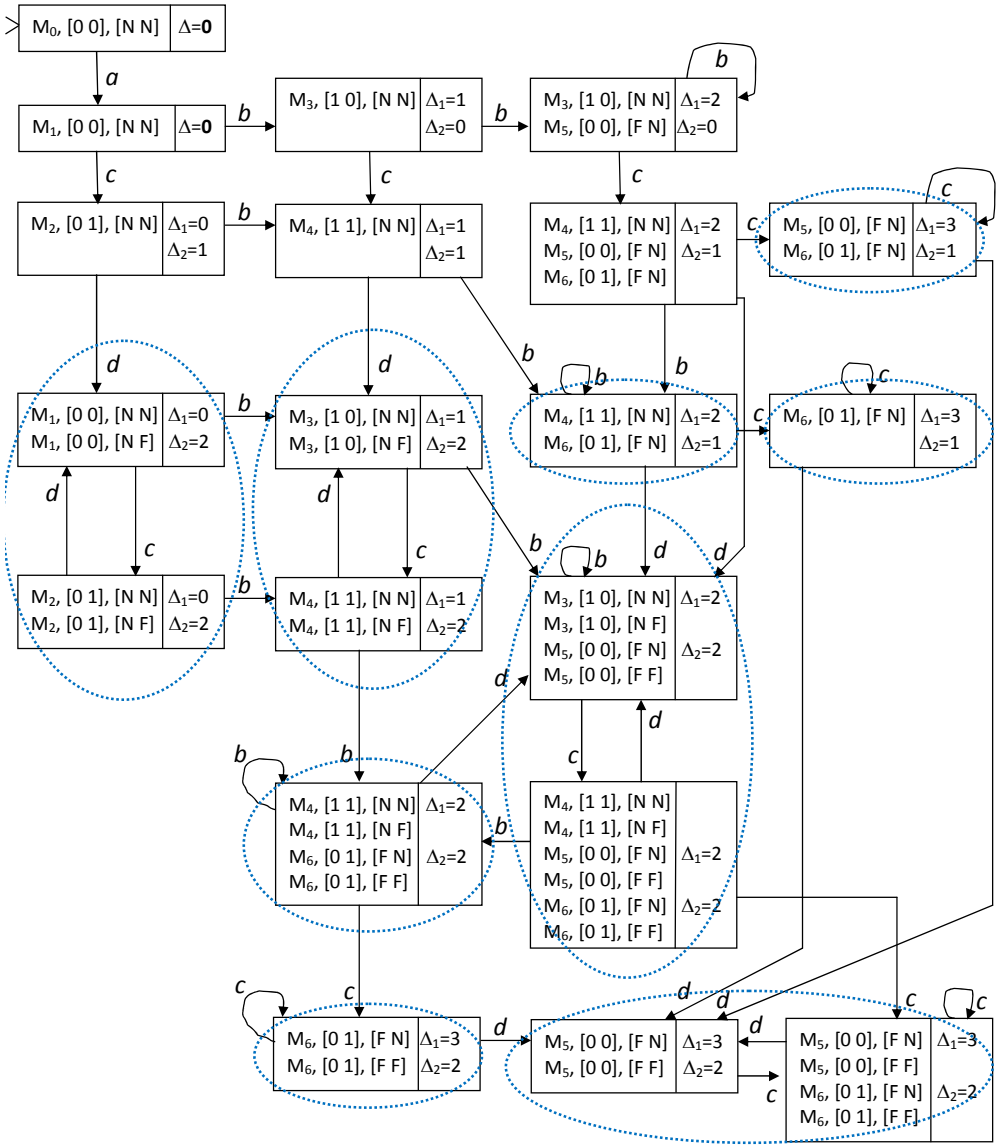

Figura 4.4: BRD della RdP in Figura 2.5 con i cicli incerti per la seconda classe di guasto.

di Robert Tarjan. Tale lavoro si basa sulla *Depth first search*(DFS)[1] a partire dal nodo iniziale del grafo. In particolare, una volta determinate tutte le componenti fortemente connesse del grafo la DFS viene applicata a ciascuna di esse. Considerato un numero di nodi N, un numero di archi A ed un dato numero di circuiti elementari C, l'algoritmo ha un tempo limite pari a

$$O((N + A)(C + 1)).$$

In seguito, si è passati ad analizzare un algoritmo di Donald B. Johnson ([16]), che si basa sul precedente lavoro di Tarjan ma è più efficiente. In particolare in tale algoritmo il tempo trascorso tra due rilevazioni consecutive di un circuito non eccede mai le dimensioni del grafo, quindi il tempo di calcolo è limitato da

$$O(N + A).$$

Entrambi gli algoritmi, in maniera più o meno efficiente, possono essere applicati per la determinazione dei cicli nel BRD, ma non forniscono tutte le informazioni necessarie per la verifica della diagnosticabilità. Infatti, non abbiamo alcuna informazione relativa a tutti quei percorsi che dal nodo iniziale del grafo portano ad uno dei nodi appartenenti al ciclo. Quindi l'informazione, legata alla verifica dell'occorrenza del guasto lungo il cammino che porta al ciclo, deve essere comunque reperita per poter ultimare l'analisi.

Sulla luce di questa analisi, si è preferita l'enumerazione esaustiva di tutti i percorsi presenti nel grafo e l'identificazione tra essi di eventuali cicli. Tale approccio, molto più macchinoso per la determinazione dei cicli, risulta più efficiente nel complesso per la verifica della diagnosticabilità. Tale algoritmo, ha un tempo limite pari a

$$O(N^L),$$

dove L è l'alfabeto della RdP.

Nel seguito, è fornito, per la prima volta, un algoritmo che riassume i passi principali per la verifica della diagnosticabilità. Tale algoritmo è basato sulle nozioni teoriche di Cabasino *et al.* ([8]). Si noti che, al fine di semplificare la notazione, si è assunto che ciascuna classe di guasto includa un'unica transizione, per cui $|T_f| = r$. L'estensione al caso più generale è banale e non viene riportata.

[1] La DFS è un algoritmo per l'esplorazione di un grafo G. Ad ogni passo si esamina un arco uscente dal vertice v esplorato più di recente tra quelli che hanno ancora archi uscenti inesplorati. Quando tutti gli archi di v sono stati esplorati, la ricerca torna indietro per esplorare gli altri archi che escono dal vertice a partire dal quale v è stato scoperto. Questo processo continua finché sono stati raggiunti tutti i vertici raggiungibili da una sorgente iniziale. Se rimangono dei vertici inesplorati, allora uno di essi viene scelto come nuova sorgente e la ricerca riparte da essa.

Algoritmo 4.4.1. [Verifica della diagnosticabilità]

1. Seleziona il nodo iniziale del BRD $d_0 = (M_0, x_0, h_0)$
2. Sia $N = d_0$ e $W = \varepsilon$, rispettivamente l'insieme dei percorsi esplorati e l'insieme delle osservazioni ad essi associate.
3. Sia $N' = \emptyset$ e $W' = \emptyset$.
4. Per tutti i percorsi $n \in N$ e le corrispondenti osservazioni $w \in W$,
  4.1. sia i l'ultimo nodo raggiunto di n.
  4.2. Per tutti gli archi $l \in L$ uscenti da i e tali che $i \xrightarrow{l} j$,
    4.2.1. sia $N' = N' \cup \{[n, j]\}$ e $W' = W' \cup \{[w, l]\}$.
5. $\forall$ ciclo $\gamma \in N$
  5.1. se γ è un ciclo incerto
    5.1.1. sia $P_o(\gamma) = \rho \in W'$ la sua proiezione osservabile
    5.1.2. $\forall$ percorso $p \in L^*$ dal nodo iniziale ad uno appartenente al suddetto ciclo, fai
      • se nell'MBRG $\exists$ una coppia di cicli $(\gamma_1,\ \gamma_2)$ tali che:
        • $P_o(\gamma_1) = P_o(\gamma_2) = \rho$;
        • $\exists$ due percorsi p_1 e $p_2 |\ P_o(p_1) = \rho$, che dal nodo iniziale dell'MBRG abilitino γ_1 e γ_2;
        • entrambi γ_2 e p_2 non contengono un guasto di T_f, mentre o γ_1 o p_1 o entrambi contengono un guasto di T_f.
          • termina l'algoritmo e dichiara non diagnosticabile la classe di guasto.
    5.1.3. siano $N' = N' \setminus \gamma$ e $W' = W' \setminus \rho$.
  5.2. Altrimenti,
    5.2.1 siano $N' = N' \setminus \gamma$ e $W' = W' \setminus \rho$.
6. Siano $N = N'$ e $W = W'$.
7. Ritorna al passo 3. ■

L'algoritmo enumera esaustivamente a partire dal nodo iniziale del BRD tutti i percorsi orientati presenti nel grafo e, per ciascuno di essi, tiene memoria della sequenza di nodi attraversati tramite l'insieme N e della proiezione osservabile associata tramite l'insieme W. Per ciascun percorso esplorato, i nodi attraversati sono memorizzati in N e le osservazioni associate a tali percorsi sono memorizzate in W. Congiuntamente alla crescita della lunghezza dei percorsi, è verificata la presenza di eventuali cicli. Se il ciclo è incerto (passo 5.1), viene verificato che sia anche indeterminato. Se la verifica va a buon fine l'algoritmo termina e dichiara non diagnosticabile la classe di guasto in esame. Altrimenti il ciclo è eliminato (passo 5.1.3), così come nel caso in cui il ciclo non fosse incerto (passo 5.2).

Esempio 4.4.2. Si consideri la RdP in Figura 2.5 il cui BRD è dato in Figura 4.2. Dall'analisi dei cicli indeterminati riportati nell'Esempio 4.4.1 concludiamo che il sistema non è diagnosticabile.

Si noti che, non appena si è trovato un ciclo indeterminato per una classe di guasto, si può concludere che il sistema non è diagnosticabile per tale classe di guasto. Al contrario, per stabilire se un sistema è diagnosticabile rispetto ad una classe di guasto è necessario esaminare tutti i cicli incerti per tale classe di guasto e mostrare che nessuno è indeterminato. ■

Capitolo 5

Implementazione del Toolbox

Sommario

In questo capitolo, verranno presentati i programmi scritti nelle loro funzionalità e differenti esempi porteranno a chiarirne il funzionamento. Verranno forniti il codice per la costruzione del Grafo di Raggiungibilità di Base, del Grafo di Raggiungibilità di Base Modificato e del Diagnosticatore di Raggiungibilità di Base per reti di Petri etichettate, i cui algoritmi sono stati forniti rispettivamente da Cabasino, Giua e Seatzu in [9] per il primo e in [8] per i restanti due. È, in aggiunta, fornito un programma per effettuare la verifica delle condizioni necessarie e sufficienti per la diagnosticabilità. L'algoritmo implementato è introdotto per la prima volta ed è basato sulle teorie introdotte sempre da da Cabasino *et al.* in [8].

5.1 Introduzione alle funzioni di MATLAB

Tutte le funzioni saranno richiamate con la seguente sintassi, comune a tutte le librerie di MATLAB:

$$[out1, out2, \cdots] = function\_name(in1, in2, \cdots)$$

Descrizione:

la sintassi di cui sopra effettua una chiamata alla funzione *function_name*, che accetta gli ingressi *in1*, *in2*, etc. e restituisce le uscite *out1*, *out2*, etc.

Si noti che per aggiungere al vocabolario di MATLAB nuove funzioni è sufficiente esprimerle in termini di funzioni esistenti, ossia creare un appropriato file di testo chiamato *M-file*. I file M possono essere *script*, contenenti una semplice sequenza di istanze, o *funzioni*, che fanno uso di proprie variabili locali (dichiarate all'interno del suo corpo) ed accettano argomenti di ingresso. Il nome di un file M inizia con un carattere alfabetico e la sua estensione è *.m*

Per maggiori informazioni, si rimanda al manuale di MATLAB e, in particolare, alla sezione *MATLAB Programming Tips* ([18]). Tale sezione, tra le varie cose, spiega come MATLAB interpreti nomi di variabili e di funzioni inserite nella *Command Window*.

In appendice A della tesi è allegato il codice completo di tutti i programmi, opportunamente commentato per eventuali delucidazioni e chiarimenti.

5.2 BRG.m

In questa sezione è presentata la funzione per il calcolo del *Grafo di Raggiungibilità di Base* (BRG) di una rete di Petri (RdP), valido strumento per effettuare la diagnosi off-line.

Data la RdP ed i parametri seguenti

INPUT:

- matrici PRE e $POST$;
- marcatura iniziale M_0;
- le classi di Guasto, definite da una matrice di celle F;
- la funzione di etichettatura L(t), definita da una matrice di celle L;
- l'alfabeto delle etichette, definito da una matrice di celle E.

la funzione restituisce

OUTPUT:

- BRG.

Richiamiamo, dunque, la funzione nella finestra dei comandi con la sintassi:

$$brg = BRG(Pre, Post, M_0, F, L, E).$$

Il grafo è dato come una matrice di celle brg, contenente tante righe quanti i nodi del grafo corrispondente. In ciascuna riga è specificato:

i) il numero del nodo;

ii) la marcatura di base associata al nodo;

iii) il vettore X-fault, soluzione del sistema del sistema di equazioni 3.1;

iv) un vettore indicante quali transizioni sono abilitate dal nodo;

v) gli archi uscenti dal nodo nel caso di RdP *non etichettate*, dichiarati con la coppia $(t; g - vettore)$;

vi) un TAG indicante se il nodo è stato esplorato o meno;

vii) un vettore indicante quali etichette sono abilitate dal nodo;

viii) gli archi uscenti dal nodo nel caso di RdP *etichettate*, dichiarati con la coppia $(\mathcal{L}(t); g - vettore)$;

Non esistono delle restrizioni particolari in termini di ordinamento delle transizioni nella stesura delle matrici di Pre e Post incidenza

F deve essere una matrice di celle di dimensione *[# classi di guasto* $\times$ *1]*, tale che nella riga i-esima ci sia un vettore riga degli indici delle transizioni non osservabili appartenenti alla classe di guasto i. L deve essere una matrice di celle di dimensione *[# etichette* $\times$ *1]*, tale che nella riga i-esima ci sia un vettore riga degli indici delle transizioni osservabili corrispondenti all'etichetta i-esima. E deve essere una matrice di celle di dimensione *[# etichette* $\times$ *1]*, tale che nella riga i-esima ci sia il simbolo associato alle transizioni osservabili corrispondenti all'etichetta i-esima.

Nel seguito, se non diversamente specificato, faremo uso delle stesse strutture.

Per illustrarne il funzionamento, faremo uso di un esempio.

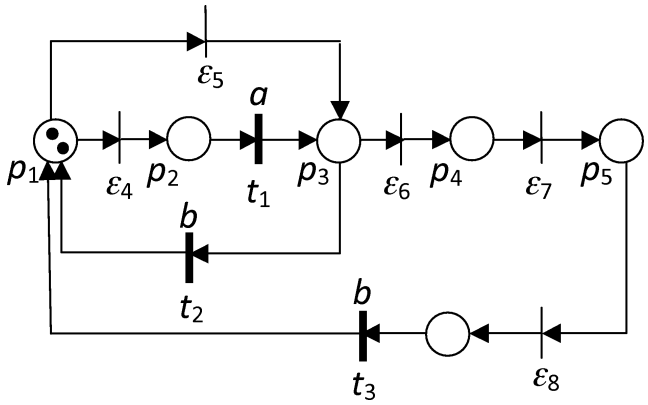

Figura 5.1: Un esempio di RdP per l'analisi del Toolbox

Esempio 5.2.1. *Consideriamo la RdP in Fig. 5.1 dove l'insieme delle transizioni osservabili è* $T_o = \{t_1, t_2, t_3\}$*, l'insieme delle transizioni non osservabili è* $T_u = \{\varepsilon_4, \varepsilon_5, \varepsilon_6, \varepsilon_7, \varepsilon_8\}$ *e l'unica classe di guasto è* $T_f = \{\varepsilon_5, \varepsilon_7\}$*. La funzione di etichettatura è tale che* $\mathcal{L}(t_1) = a$ *mentre* $\mathcal{L}(t_2) = \mathcal{L}(t_3) = b$*.*

Dichiariamo innanzitutto gli ingressi della funzione nella maniera che segue:

$$Pre = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad Post = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_0 = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$$

$$F = \begin{Bmatrix} 5 & 7 \end{Bmatrix}, \quad L = \begin{Bmatrix} 1 \\ 2 & 3 \end{Bmatrix}, \quad E = \begin{Bmatrix} a \\ b \end{Bmatrix}$$

Fatto ciò, non ci resta che richiamare correttamente la funzione BRG*. Come si vede facilmente in Figura 5.2.* ■

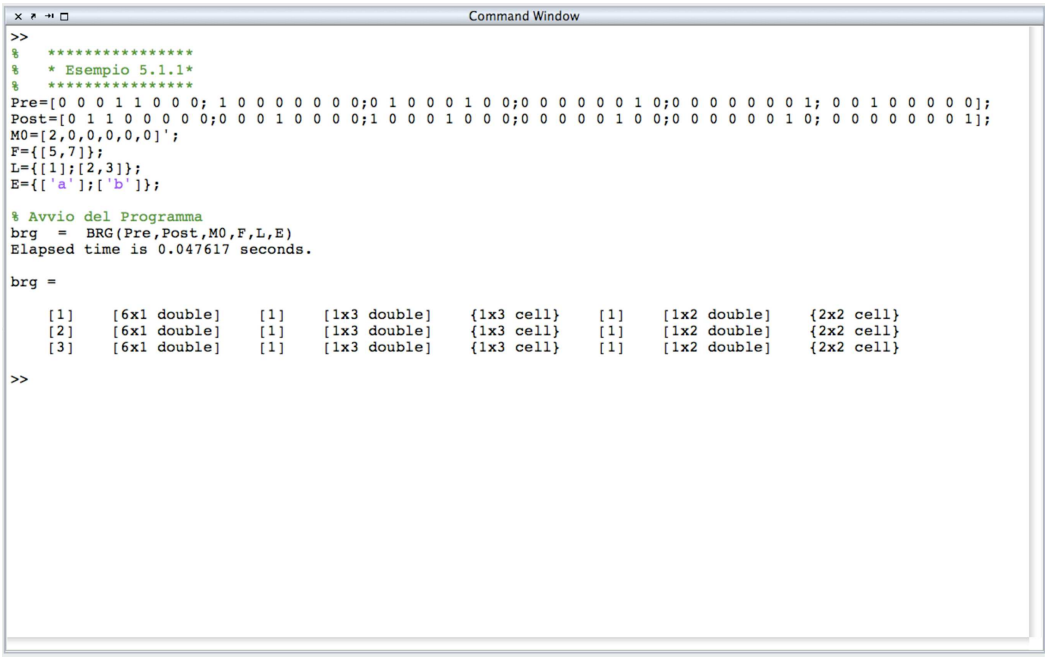

Figura 5.2: Command Window di MATLAB della funzione BRG in riferimento all'Esempio 5.2.1

5.3 showBRG.m

Poichè, probabilmente, una matrice dati di tale struttura può risultare di difficile lettura, è sembrato opportuno scrivere un programma che potesse estrapolarne i dati in maniera veloce e renderli accessibili all'utente.

Tale funzione ha: INPUT:

- BRG.

OUTPUT:

- visualizzazione testuale del BRG.

Richiamando, dunque, la funzione *showBRG.m* nella maniera che segue

$$showBRG(brg),$$

viene ottenuto come risultato la schermata di Figura 5.3, che chiaramente coincide con il BRG di Figura 5.4, i cui dati vengono esplicitati nelle tabelle 5.1 e 5.2.

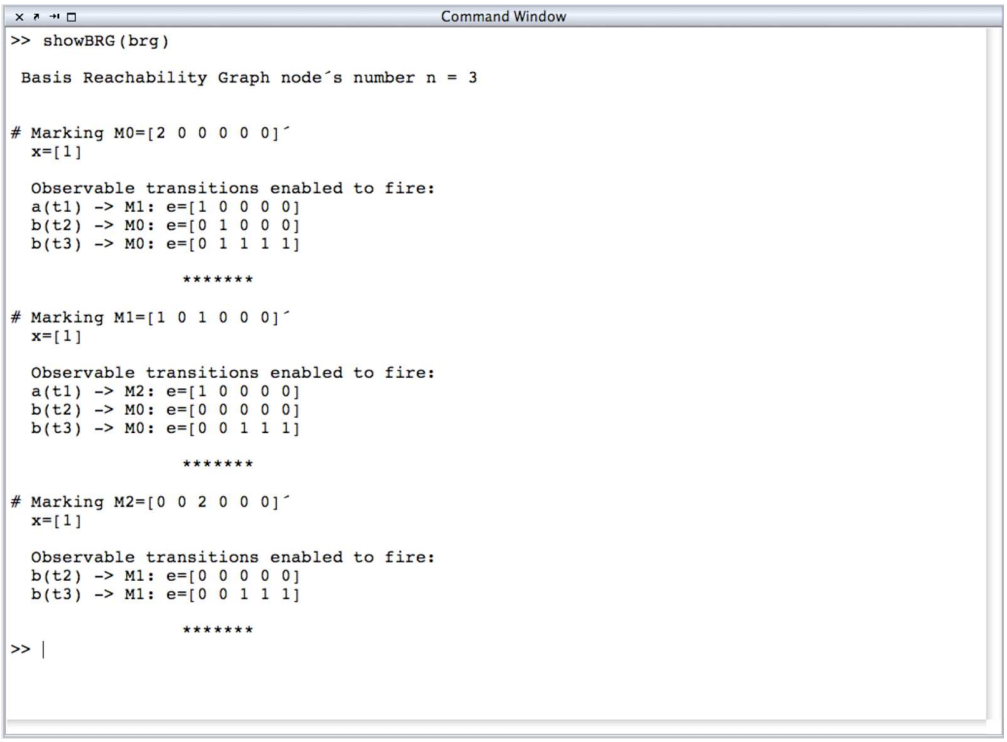

Figura 5.3: Command Window di MATLAB della funzione showBRG in riferimento all'Esempio 5.4.1

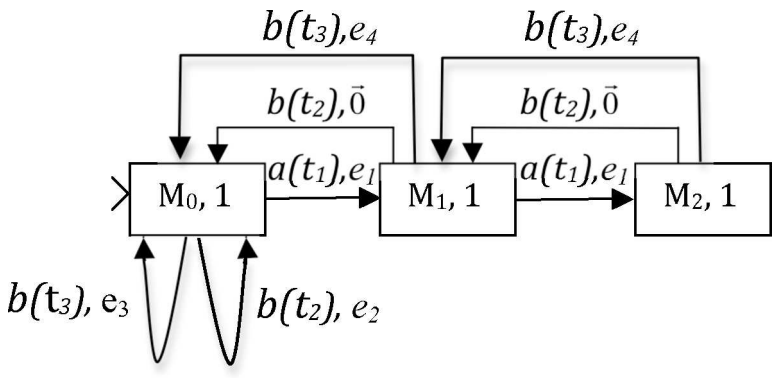

Figura 5.4: BRG della RdP dell'Esempio 5.2.1

	ε_4	ε_5	ε_6	ε_7	ε_8
e_1	1	0	0	0	0
e_2	0	1	0	0	0
e_3	0	1	1	1	1
e_4	0	0	1	1	1

Tabella 5.1: Gli e-vettori del BRG in Figura 5.4.

M_0	2	0	0	0	0	0
M_1	1	0	1	0	0	0
M_2	0	0	2	0	0	0

Tabella 5.2: Le marcature di base del BRG in Figura 5.4.

5.4 MBRG.m

In questa sezione è presentata la funzione per il calcolo del *Grafo di Raggiungibilità di Base Modificato* (MBRG) di una RdP, che verrà utilizzato per l'analisi della diagnosticabilità.

Data la RdP ed i parametri seguenti

INPUT:

- matrici PRE e $POST$;
- marcatura iniziale M_0;
- le classi di Guasto, definite da una matrice di celle F;
- la funzione di etichettatura L(t), definita da una matrice di celle L;
- l'alfabeto delle etichette, definito da una matrice di celle E.

la funzione restituisce

OUTPUT:

- MBRG.

Richiamiamo, dunque, la funzione nella finestra dei comandi con la sintassi:

$$mbrg = MBRG(Pre, Post, M_0, F, L, E).$$

La matrice di celle $mbrg$ ottenuta, ha tante righe quanti i nodi del grafo corrispondente e in ciascuna di esse è specificato:

i) il numero del nodo;

ii) la marcatura di base associata al nodo;

iii) il vettore X-fault, soluzione del sistema del sistema di equazioni 3.1;

iv) un vettore indicante quali transizioni osservabili e quali non osservabili di guasto sono abilitate dal nodo;

v) gli archi uscenti dal nodo nel caso di RdP *non etichettate*, dichiarati con la coppia $(t, g-vettore)$ o $(\varepsilon, g-vettore)$;

vi) un TAG indicante se il nodo è stato esplorato o meno;

vii) un vettore indicante quali etichette e classi di guasto sono abilitate dal nodo;

viii) gli archi uscenti dal nodo nel caso di RdP *etichettate*, dichiarati con la coppia $(\mathcal{L}(t), g-vettore)$ o $(\varepsilon, g-vettore)$.

Un esempio ne chiarirà l'utilizzo.

Esempio 5.4.1. *Consideriamo la RdP in Fig. 5.1 trattata nell'Esempio 5.2.1. Gli ingressi della funzione rimangono gli stessi del caso precedente.Per ottenere l'uscita desiderata, non ci resta che richiamare la funzione* MBRG*, come si vede facilmente in Figura 5.5* ■

5.5 showMBRG.m

Analogamente alla costruzione del BRG, è stato scritto un programma che estrae i dati e li mette a video i maniera testuale.

Tale funzione ha:

INPUT:

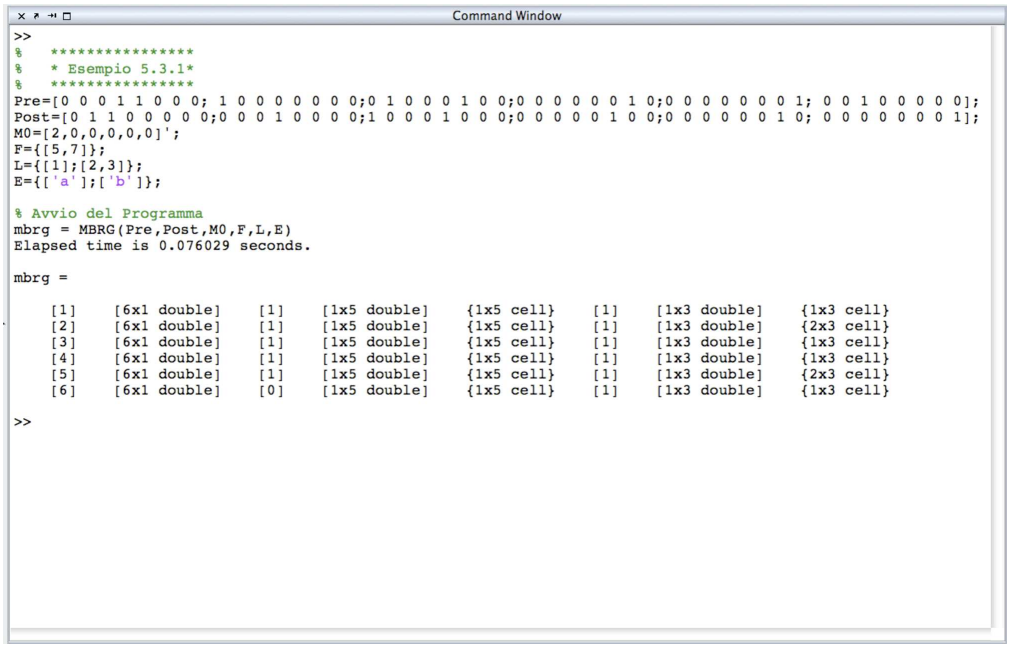

Figura 5.5: Command Window di MATLAB della funzione MBRG in riferimento all'Esempio 5.4.1

- MBRG.

OUTPUT:

- visualizzazione testuale dell'MBRG.

Richiamando, dunque, la funzione *showBRG.m* nella maniera che segue

$$showMBRG(mbrg),$$

viene ottenuto come risultato la schermata di Figura 5.6, che chiaramente coincide con l'MBRG di Figura 5.7, i cui dati vengono esplicitati in Tabella 5.3 e 5.4.

5.6 diagnosis.m

In questa sezione è presentata la funzione per il calcolo della del *diagnosi on-line* di una RdP.

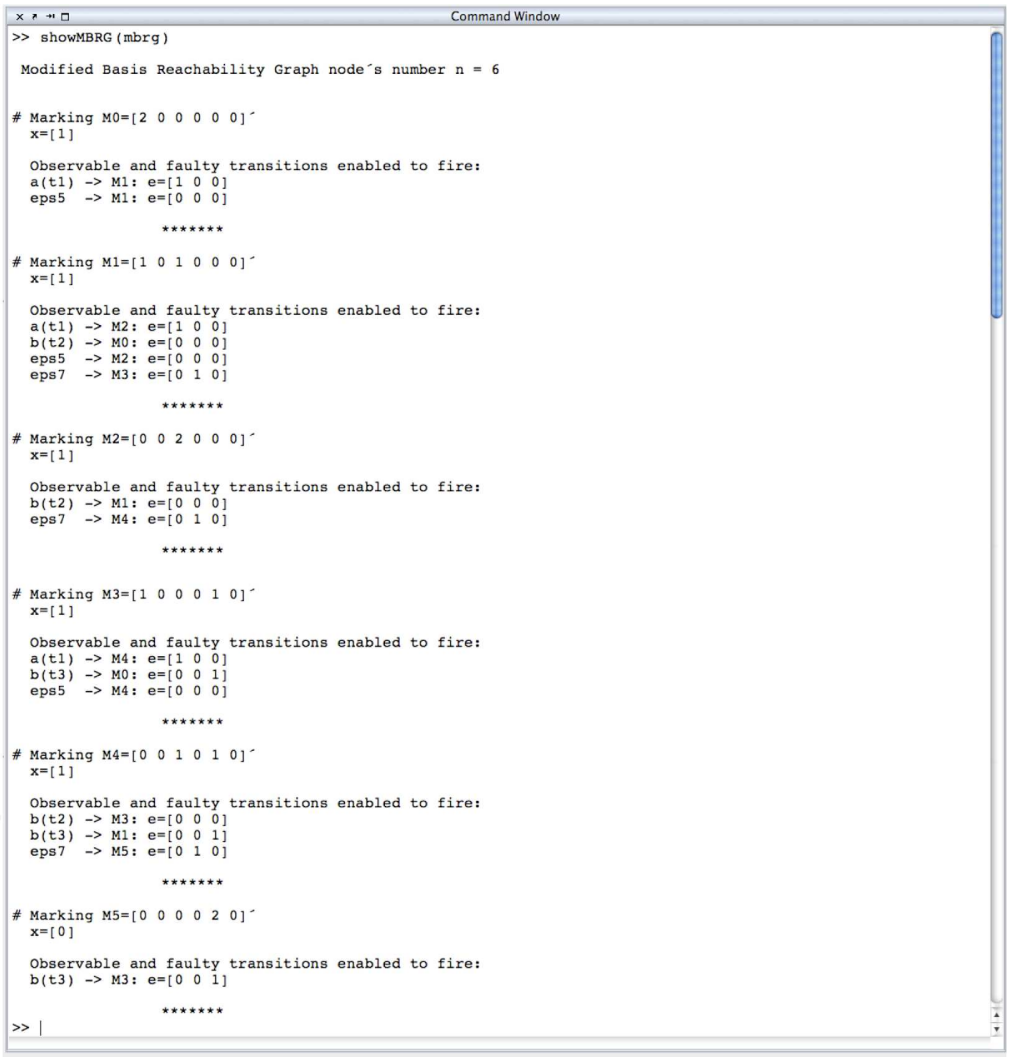

Figura 5.6: Command Window di MATLAB della funzione showMBRG in riferimento all'Esempio 5.4.1

	ε_4	ε_6	ε_8
e_1	1	0	0
e_2	0	1	0
e_3	0	0	1

Tabella 5.3: Gli e-vettori modificati dell'MBRG in Figura 5.7.

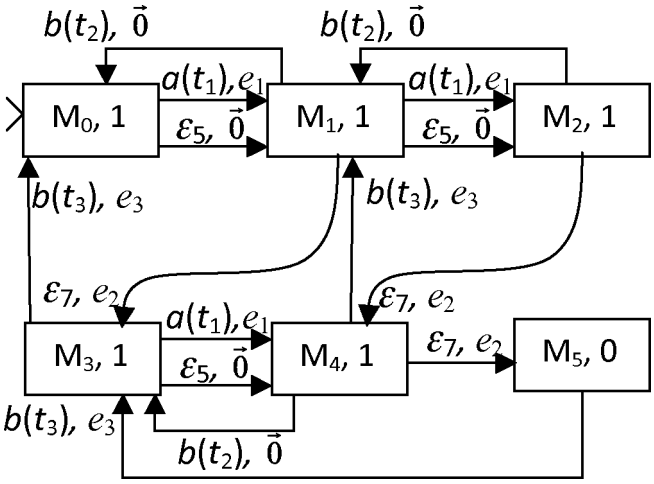

Figura 5.7: MBRG della RdP dell'Esempio 5.2.1

M_0	$[\ 2\ 0\ 0\ 0\ 0\ 0]^T$
M_1	$[\ 1\ 0\ 1\ 0\ 0\ 0]^T$
M_2	$[\ 0\ 0\ 2\ 0\ 0\ 0]^T$
M_3	$[\ 1\ 0\ 0\ 0\ 1\ 0]^T$
M_4	$[\ 0\ 0\ 1\ 0\ 1\ 0]^T$
M_5	$[\ 0\ 0\ 0\ 0\ 2\ 0]^T$

Tabella 5.4: Le marcature dell'MBRG in Figura 5.7.

Data la RdP ed i parametri seguenti

INPUT:

- matrici PRE e $POST$;
- marcatura iniziale M_0;
- parola osservata w;
- classi di Guasto, definite da una matrice di celle F;
- funzione di etichettatura L(t), definita da una matrice di celle L;
- alfabeto delle etichette, definito da una matrice di celle E.

la funzione restituisce

OUTPUT:

- visualizzazione testuale dell'evoluzione dello stato di diagnosi.

Richiamiamo, dunque, la funzione nella finestra dei comandi con la sintassi:

$$diagnosis(Pre, Post, M_0, w, F, L, E).$$

La funzione genera localmente una matrice di celle D, contenente tante righe quanto il numero di etichette componenti la parola osservata più uno. In aggiunta, visualizza a schermo il progressivo evolversi dello stato di diagnosi del sistema. In ciascuna riga è specificato:

i) il suffisso della parola in analisi;

ii) lo stato di diagnosi del sistema in riferimento a ciascuna classe di guasto.

Indicando con $|w|$ cardinalità parola osservata, si definisce w come un vettore di dimensione [1 $\times$ |w|], tale che nella colonne i-esima ci sia la i-esima etichetta scattata in ordine temporale dall'inizio dell'osservazione.

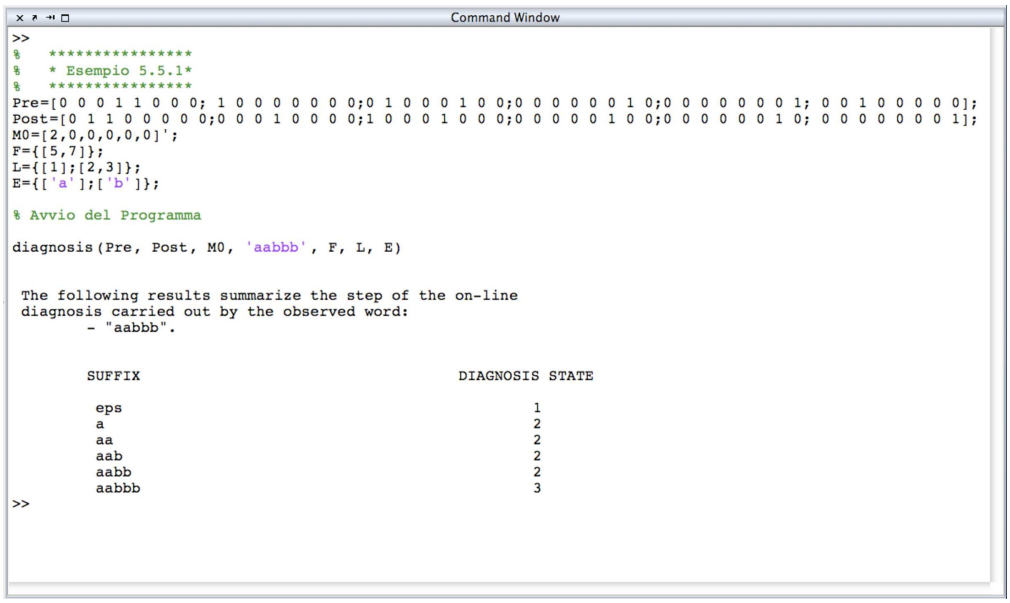

Figura 5.8: Command Window di MATLAB della funzione *diagnosis* in riferimento all'Esempio 5.6

Esempio 5.6.1. *Si consideri nuovamente il sistema di RdP in Figura 5.1, dove* $T_f = \{\varepsilon_5, \varepsilon_7\}$. *Sia* $w = aabbb$. *Si ricava*

$\mathcal{M}(w) = \{(M_0, [2\,1\,1\,1\,1]^T), (M_0, [2\,1\,0\,0\,0]^T), (M_0, [2\,1\,2\,2\,2]^T), (M_0, [2\,1\,1\,1\,1]^T),$
$(M_0, [2\,1\,3\,3\,3]^T), (M_0, [2\,1\,2\,2\,2]^T)\},$

dove $M_0 = [2\,0\,0\,0\,0\,0]^T$ *è la marcatura iniziale. In questo caso* $\Delta(w, T_f) = 3$, *esattamente come mostrato dalla schermata di MATLAB in figura 5.8. Infatti,* $y(\varepsilon_5) > 0$ *per tutte le (M,y)* $\in \mathcal{M}$*(aabbb).* ■

5.7 BRD.m

In questa sezione è presentata la funzione per il calcolo del *Diagnosticatore di Raggiungibilità di Base* (BRD) di una RdP, che verrà utilizzato per l'analisi della diagnosticabilità.

Data la RdP ed i parametri seguenti

INPUT:

- BRG
- matrici PRE e $POST$;
- marcatura iniziale M_0;
- parola osservata w;
- classi di Guasto, definite da una matrice di celle F;
- funzione di etichettatura L(t), definita da una matrice di celle L;
- alfabeto delle etichette, definito da una matrice di celle E.

la funzione restituisce

OUTPUT:

- BRD

Dunque, richiamiamo la funzione nella finestra dei comandi con la sintassi:

$$brd = BRD(brg, Pre, Post, M_0, F, L, E).$$

La funzione restituisce una matrice di celle brd, contenente tante righe quanti i nodi del grafo corrispondente. In ciascuna riga è specificato:

i) il numero del nodo;

ii) le marcature di base associate al nodo, il vettore X-fault, soluzione del sistema del sistema di equazioni 3.1 e la quantità $h \in \{N, F\}^{|T_f|}$;

iii) un vettore indicante lo stato di diagnosi con riferimento alle classi di guasto;

iv) gli archi etichettati uscenti dal nodo ed il corrispondente nodo raggiunto;

v) gli indici relativi alle marcature di base del nodo;

vi) un TAG indicante se il nodo è stato esplorato o meno.

Un esempio, ne chiarirà l'utilizzo.

Esempio 5.7.1. *Consideriamo la RdP in Figura 5.1 trattata nell'Esempio 5.2.1. Gli ingressi della funzione rimangono gli stessi del caso precedente, così, per ottenere l'uscita desiderata, non ci resta che richiamare la funzione* BRD, *come si vede facilmente in Figura 5.9, ed analizzarne l'uscita.* ■

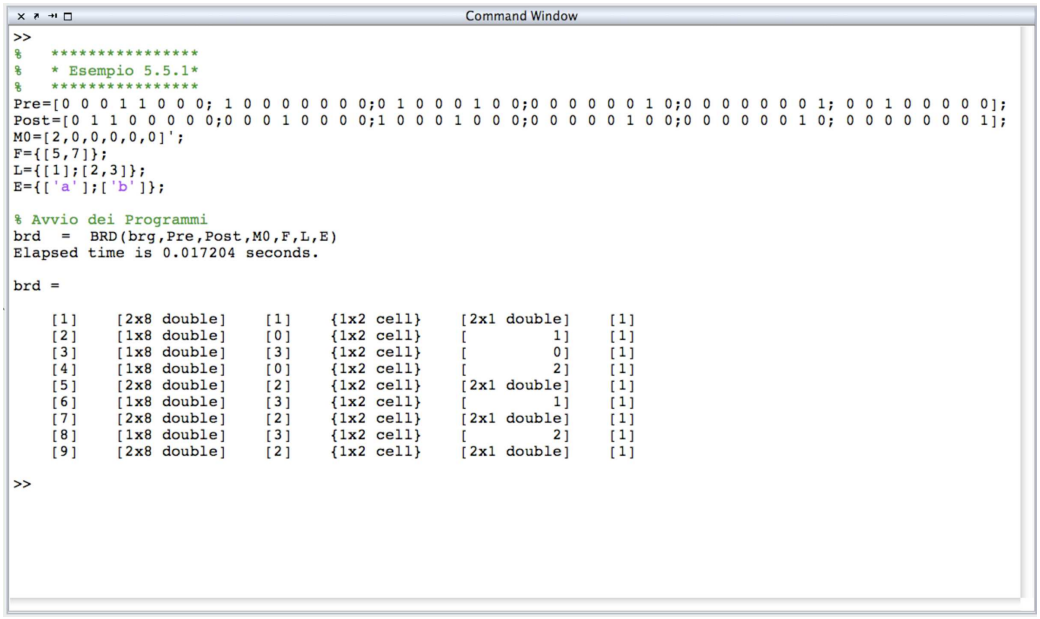

Figura 5.9: Command Window di MATLAB della funzione BRD in riferimento all'Esempio 5.7.1

5.8 showBRD.m

Analogamente alla costruzione di BRG ed MBRG, è stato scritto un programma che estrae i dati e li mette a video i maniera testuale.

Tale funzione ha: INPUT:

- BRD.

OUTPUT:

- visualizzazione testuale del BRD.

Richiamando, dunque, la funzione *showBRD.m* nella maniera che segue

$$showBRD(brd),$$

viene ottenuto come risultato la schermata in Figura 5.11 e 5.12, che chiaramente coincide con il BRD di Figura 5.10, i cui dati vengono esplicitati in Tabella 5.4.

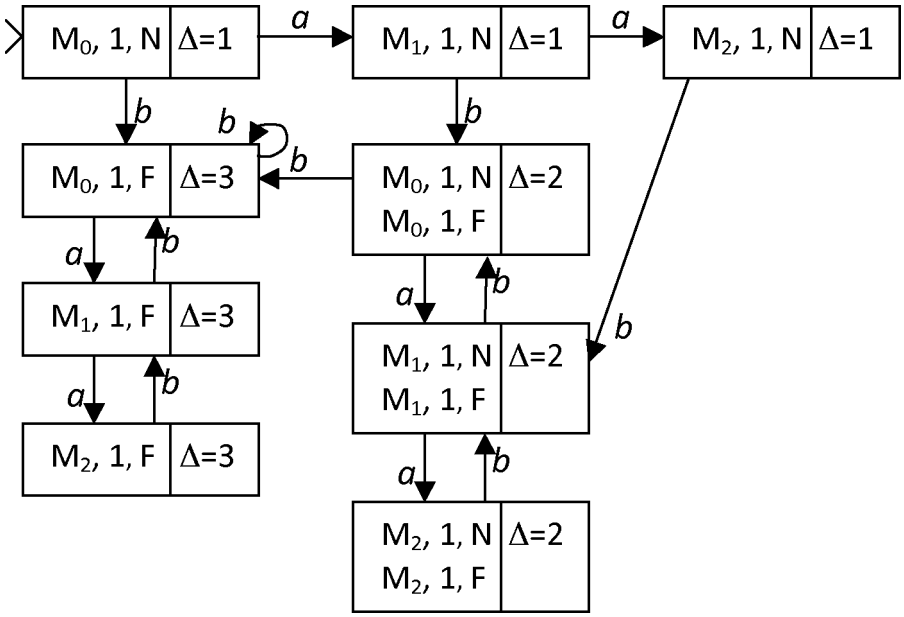

Figura 5.10: BRD della RdP dell'Esempio 5.2.1

5.9 diagnosability.m

Questa funzione verifica le condizioni necessarie e sufficienti per la diagnosticabilità di sistemi di RdP limitate. Il metodo utilizzato per tale accertamento è basato sull'analisi di due dei grafi precedentemente introdotti, il BRD e l'M-BRG. Nel particolare, ci preoccuperemo di individuare la possibile presenza di cicli indeterminati nel BRD.

Per tale funzione:

INPUT:

- MBRG;
- BRD.

OUTPUT:

- visualizzazione testuale dell'analisi di verifica della diagnosticabilità.

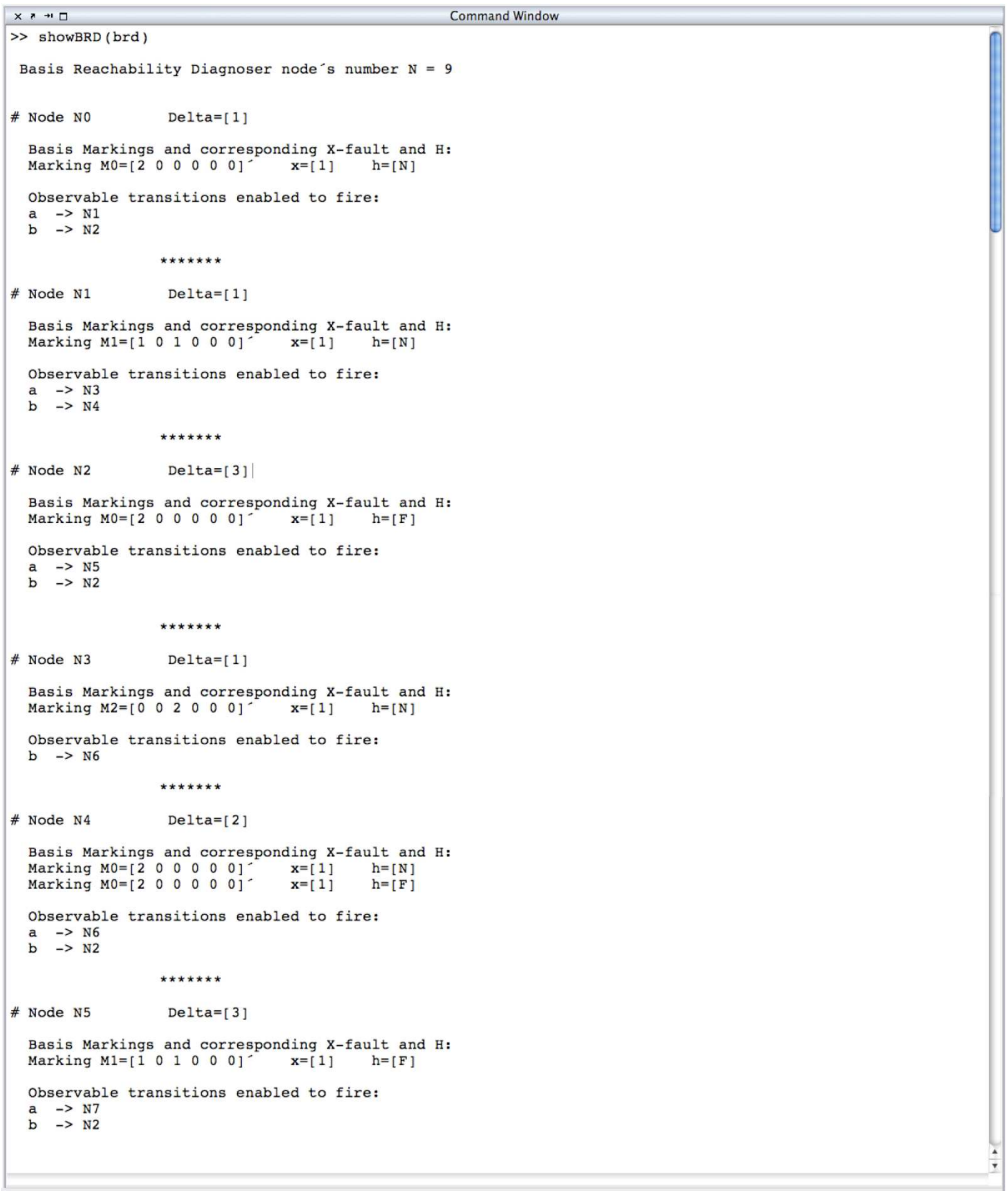

Figura 5.11: Command Window [1/2] di MATLAB della funzione showBRD in riferimento all'Esempio 5.7.1

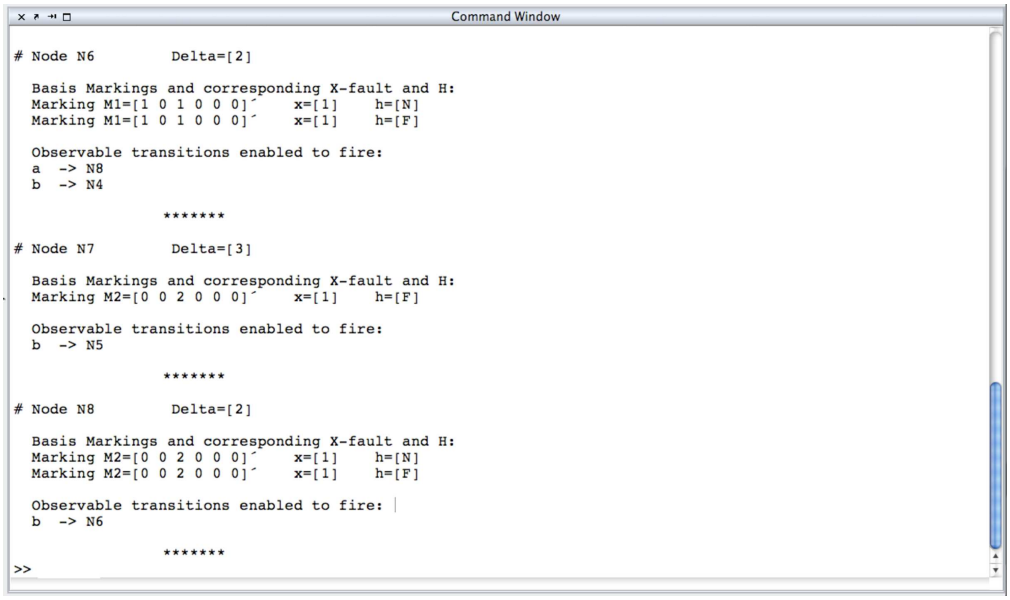

Figura 5.12: Command Window [2/2] di MATLAB della funzione showBRD in riferimento all'Esempio 5.7.1

Richiamiamo la funzione nella finestra dei comandi con la sintassi:

$$diagnosability(mbrg, brd).$$

La funzione visualizza a schermo quali classi di guasto sono diagnosticabili e quali no. Per ciascuna classe di guasto non diagnosticabile, viene visualizzato un ciclo indeterminato ed il relativo percorso.

Esempio 5.9.1. *Consideriamo il BRD in Figura 5.10 della RdP in Figura 5.1 trattata nell'Esempio 5.2.1. Chiamata correttamente la funzione* diagnosability, *si conclude che l'unica classe di guasto non è diagnosticabile, per la presenza di un ciclo indeterminato. In Figura 5.14 ne è mostrato il risultato, coincidente con il grafo in Figura 5.13.* ■

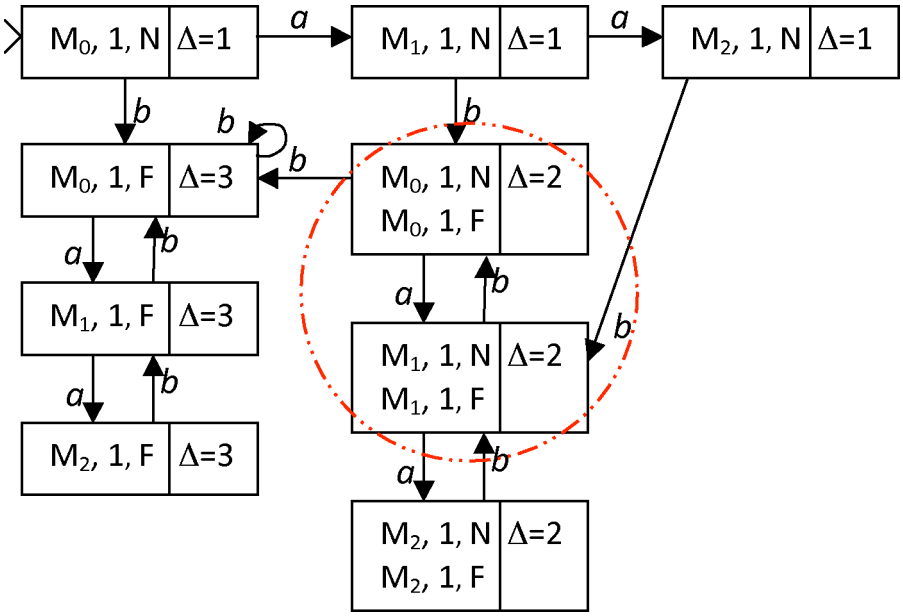

Figura 5.13: Visualizzazione del ciclo indeterminato proposto da MATLAB per l'Esempio 5.9.1

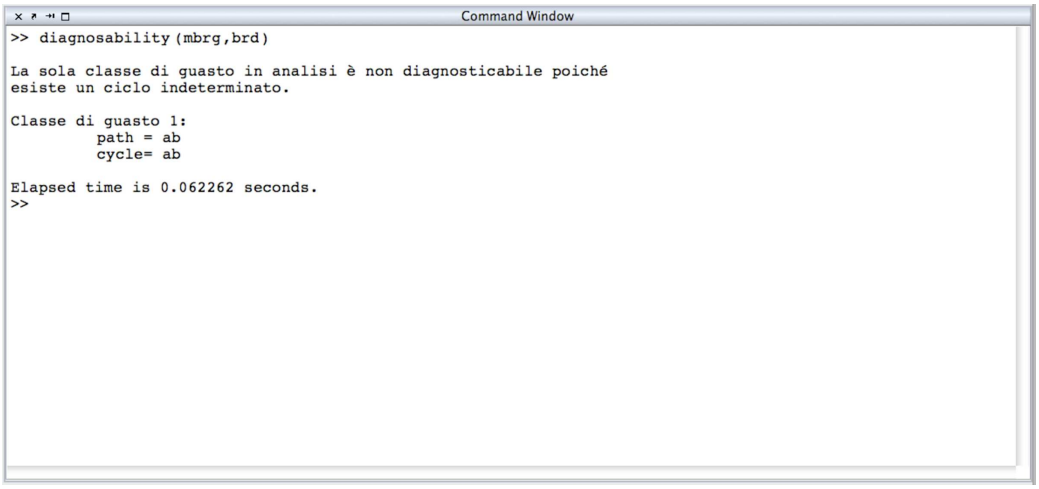

Figura 5.14: Command Window di MATLAB della funzione diagnosability in riferimento all'Esempio 5.9.1

Capitolo 6

Risultati numerici

Sommario

In questo capitolo verranno presentati alcuni risultati numerici ottenuti applicando alcuni programmi del toolbox sviluppato ad un modello parametrico di sistema manifatturiero, descritto formalmente dalla corrispondente *rete di Petri* (RdP)

6.1 Modello fisico considerato

Il modello fisico considerato descrive una particolare famiglia di sistemi manifatturieri. Il sistema è composto da due gruppi di lavoro perfettamente simmetrici per operatività e finalità, atti, quindi, alla produzione dello stesso prodotto. Ciascun gruppo è caratterizzato da m linee di produzione, su cui vengono pilotate m differenti parti dello stesso componente. Ciascuna linea di produzione effettua l differenti operazioni, modellate con l transizioni regolari ε_i ($i = 1, \ldots, l$), prima di poters fornire ciascuno dei pezzi. Sono presenti $m - 1$ transizioni di guasto, rappresentate da transizioni non osservabili f_i ($i = 1, \ldots, m - 1$). Si noti che il guasto f_i comporta lo spostamento accidentale di un pezzo dalla linea di produzione i a quella $i + 1$. La terminazione della i-esima catena di l operazioni è modellata con la transizione osservabile etichettata a_i. Questo comporta che, nonostante tutte le parti del componente risultino lavorate correttamente, ossia siano presenti all'appello $2m$ pezzi finiti, alcune di esse abbiano subito un trattamento anziché un altro, compromettendo la funzionalità del prodotto assemblato. Affin-

ché si possa verificare la diagnosticabilità del sistema, si è pensato di etichettare in due differenti maniere le transizioni osservabili. Nello specifico, nel caso in cui la prima transizione del primo gruppo di lavoro produca la stessa osservazione della seconda, si verificherebbe che il guasto non sia diagnosticabile.

Nel particolare, il sistema, nel suo complesso è, quindi, caratterizzato da tre parametri:

1. $2m$, il numero totale di linee di produzione;
2. l, il numero di operazioni che devono essere effettuate su ciascun componente del prodotto;
3. d, una variabile binaria che comporta alcune caratteristiche nella funzione di l'etichettatura. Nel particolare:
   $$d = \begin{cases} 1, \text{ se il sistema è diagnosticabile} \\ 0, \text{ in caso contrario} \end{cases}$$

Tale modello è rappresentato dalla RdP in Figura 6.1.

6.2 Simulazioni numeriche

In questa sezione presenteremo alcuni risultati numerici del Tool sviluppato e, nel particolare, dei programmi per la determinazione del *Grafo di Raggiungibilità di Base* (BRG), del *Grafo di Raggiungibilità di Base Modificato* (MBRG), del *Diagnosticatore di Raggiungibilità di Base* (BRD) e delle condizioni necessarie e sufficienti per la diagnosticabilità dei guasti. I risultati, ottenuti per i diversi valori dei parametri *m*, *l* e *d*, sono rappresentati nelle Tabelle 6.1 e 6.2 e nelle Figure 6.3 e 6.4.

Si noti che, per semplicità, si è supposto che tutte le transizioni di guasto appartenessero alla stessa classe di guasto.

Le tabelle sono strutturate nella maniera che segue:

- le colonne 1 e 2 mostrano il valore dei parametri m ed l;
- le colonne 3 e 4 indicano il numero di nodi del grafo di raggiungibilità della RdP, quindi, il numero di stati dell'automa a stati finiti modellante il sistema, ed il tempo, espresso in secondi, necessario per costruirlo;

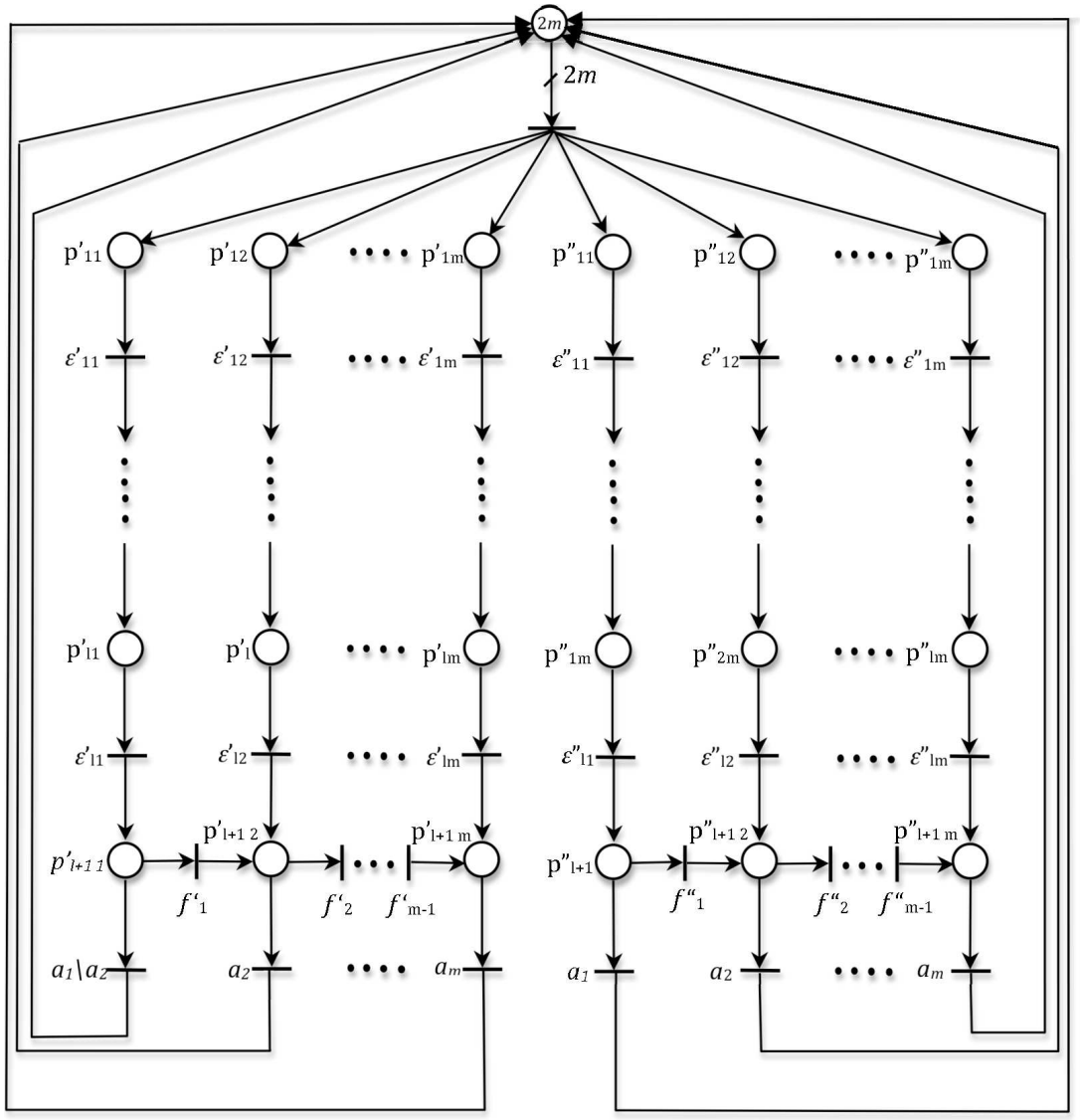

Figura 6.1: La RdP del modello fisico considerato.

- le colonne 5 e 6 indicano rispettivamente la cardinalità del BRG ed il tempo, espresso in secondi, necessario per costruirlo;
- le colonne 7 e 8 indicano rispettivamente la cardinalità dell'MBRG ed il tempo, espresso in secondi, necessario per costruirlo;
- le colonne 9 e 10 indicano rispettivamente la cardinalità del BRD ed il tempo, espresso in secondi, necessario per costruirlo; si noti che tale valore è comprensivo del tempo richiesto per la costruzione del BRG, di cui si fa uso;
- la colonna 11, l'ultima, indica il tempo, espresso in secondi, necessario per la determinazione delle condizioni necessarie e sufficienti per la diagnosticabilità; si noti che tale valore è comprensivo del tempo richiesto per la costruzione di BRD ed MBRG, di cui si fa uso.

Si osservi che alcune celle non presentano valori numerici. Esse presenteranno le voci:

- *o.t.* (*out of time*), nel caso in cui il corrispondente programma non possa essere terminato entro il limite prefissato delle 3 ore;
- *n.c.* (*not computable*), nel caso in cui il corrispondente valore non possa essere calcolato. Nel caso in cui per un certo set di parametri la costruzione del BRG fosse fuori tempo limite, allora il corrispondente BRD, che necessita di tale grafo, non potrebbe essere valutato;
- *o.m.* (*out of memory*), nel caso in cui il corrispondente valore non possa essere valutato, poiché terminata la memoria virtuale del calcolatore.

Tutte le simulazioni sono state eseguite con un PC Intel® Core™ Quad con una frequenza di clock pari a 2.40 GHz e RAM 2 GB DDR2.

m	l	$\|R\|$	t_R [s]	$\|BRG\|$	t_{BRG} [s]	$\|MBRG\|$	t_{MBRG} [s]	$\|BRD\|$	$t_{BRD}[s]$	$t_{diag}[s]$
2	1	121	0.188	16	0.061	36	0.140	23	0.157	0.260
2	2	362	1.273	16	0.063	36	0.146	23	0.165	0.396
2	3	841	7.215	16	0.068	36	0.159	23	0.172	0.415
2	4	1681	30.8	16	0.076	36	0.176	23	0.181	0.443
3	1	2025	71.1	64	0.453	484	6.793	67	2.978	14.8
3	2	10000	2130.3	64	0.519	484	7.105	67	3.063	15.0
3	3	*o.t.*	*o.t.*	64	0.586	484	7.628	67	3.143	15.6
3	4	*o.t.*	*o.t.*	64	0.661	484	7.933	67	3.254	16.1
4	1	*o.m.*	*o.m.*	256	4.574	8100	1703.9	199	101.1	*o.t.*
4	2	*o.m.*	*o.m.*	256	5.169	8100	1720.0	199	102.6	*o.t.*
4	3	*o.m.*	*o.m.*	256	5.788	8100	1751.5	199	104.7	*o.t.*
4	4	*o.m.*	*o.m.*	256	6.439	8100	1770.4	199	106.1	*o.t.*
5	1	*o.m.*	*o.m.*	1024	66.9	*o.t.*	*o.t.*	595	4379.7	*n.c.*
5	2	*o.m.*	*o.m.*	1024	73.7	*o.t.*	*o.t.*	595	4501.5	*n.c.*
5	3	*o.m.*	*o.m.*	1024	76.4	*o.t.*	*o.t.*	595	4589.8	*n.c.*
5	4	*o.m.*	*o.m.*	1024	80.8	*o.t.*	*o.t.*	595	4672.9	*n.c.*

Tabella 6.1: Risultati numerici per la rete in Figura 6.1 nel caso in cui d=0.

m	l	$\|R\|$	t_R [s]	$\|BRG\|$	t_{BRG} [s]	$\|MBRG\|$	t_{MBRG} [s]	$\|BRD\|$	$t_{BRD}[s]$	$t_{diag}[s]$
2	1	121	0.143	16	0.059	36	0.132	19	0.116	0.256
2	2	361	1.273	16	0.063	36	0.147	19	0.120	0.270
2	3	841	7.251	16	0.070	36	0.160	19	0.127	0.290
2	4	1681	30.5	16	0.081	36	0.174	19	0.140	0.318
3	1	2025	70.9	64	0.455	484	6.802	55	2.033	8.873
3	2	10000	2145.8	64	0.519	484	7.106	55	2.109	9.253
3	3	*o.t.*	*o.t.*	64	0.590	484	7.621	55	2.187	9.846
3	4	*o.t.*	*o.t.*	64	0.665	484	7.947	55	2.285	10.2
4	1	*o.m.*	*o.m.*	256	4.568	8100	1707.1	163	72.6	*o.t.*
4	2	*o.m.*	*o.m.*	256	5.104	8100	1723.5	163	73.8	*o.t.*
4	3	*o.m.*	*o.m.*	256	5.750	8100	1748.2	163	75.5	*o.t.*
4	4	*o.m.*	*o.m.*	256	6.424	8100	1777.4	163	76.5	*o.t.*
5	1	*o.m.*	*o.m.*	1024	68.2	*o.t.*	*o.t.*	487	3347.7	*n.c.*
5	2	*o.m.*	*o.m.*	1024	73.9	*o.t.*	*o.t.*	487	3414.9	*n.c.*
5	3	*o.m.*	*o.m.*	1024	76.6	*o.t.*	*o.t.*	487	3484.3	*n.c.*
5	4	*o.m.*	*o.m.*	1024	80.5	*o.t.*	*o.t.*	487	3544.3	*n.c.*

Tabella 6.2: Risultati numerici per la rete in Figura 6.1 nel caso in cui d=1.

Esiste, come già è stato detto, un toolbox per la diagnosticabilità di automi a stati finiti, che si basa sul *Diagnoser approach* di Lafortune, con cui non è stato fatto un confronto specifico per brevità. In Figura 6.2 è stata, però riportata la cardinalità dell'automa corrispondente al sistema di RdP analizzato. Si noti come alcuni valori del grafo di raggiungibilità, determinati per $m = 3$, non siano presenti in tabella poiché determinati oltre il tempo limite prefissato, ma vengano riportati per completezza e per una maggiore comprensione in Figura 6.2.

Le Tabelle 6.1 e 6.2 mostrano come le dimensioni del grafo di raggiungibilità (R) crescano notevolmente con le dimensioni della rete, ossia con m ed l. Per alcuni casi, si vedano per esempio le reti costruite per $m = 3$ ed $l > 2$, non è più possibile definire l'automa, quindi, conseguentemente, nemmeno effettuare lo studio della diagnosticabilità, cosa che il toolbox fornito effettua con discreta velocità.

Questo inizia a porre dei limiti all'approccio con gli automi, di cui è nota l'esplosione dello spazio di stato e dei relativi tempi di calcolo al crescere della complessità del modello. Da un punto di vista computazionale, il toolbox basato sulle RdP risulta migliore di quello basato sugli automi per via del concetto di *Marcatura di Base*, che permette di descrivere lo spazio di stato in maniera compatta. Al contrario, l'approccio con gli automi è basato sull'enumerazione esaustiva di tutti gli stati raggiungibili.

Al contrario, le cardinalità di BRG, MBRG e BRD sono invarianti rispetto alla lunghezza della linea di produzione e risultano essere contenute e, comunque, sempre inferiori rispetto ad R. Analogo discorso per i necessari tempi di calcolo, che crescono all'aumentare del numero di linee di produzione, ma che sostanzialmente variano in maniera poco significativa con il crescere della lunghezza della rete.

Per stabilire se un sistema è diagnosticabile rispetto ad una classe di guasto è necessario esaminare tutti i cicli incerti per tale classe di guasto e mostrare che nessuno di essi è indeterminato. Al contrario, quando un sistema è non diagnosticabile rispetto ad una classe di guasto è sufficiente trovare un ciclo indeterminato per tale classe di guasto.

Di conseguenza, in particolar modo per sistemi di dimensioni elevate, ci si aspetta che t_{diag} sia notevolmente superiore nel caso in cui d sia pari a 1 rispetto al caso in cui d sia pari a 0. Nell'esempio da noi trattato, la differenza tra i due casi è riscontrabile ma non nella misura aspettata. Infatti, a causa della particolare struttura della rete stessa, il numero di percorsi da esplorare nel caso in cui d sia pari a 1 risulta sensibilmente ridotto per via dell'alto numero di nodi con stato di diagnosi pari a 3.

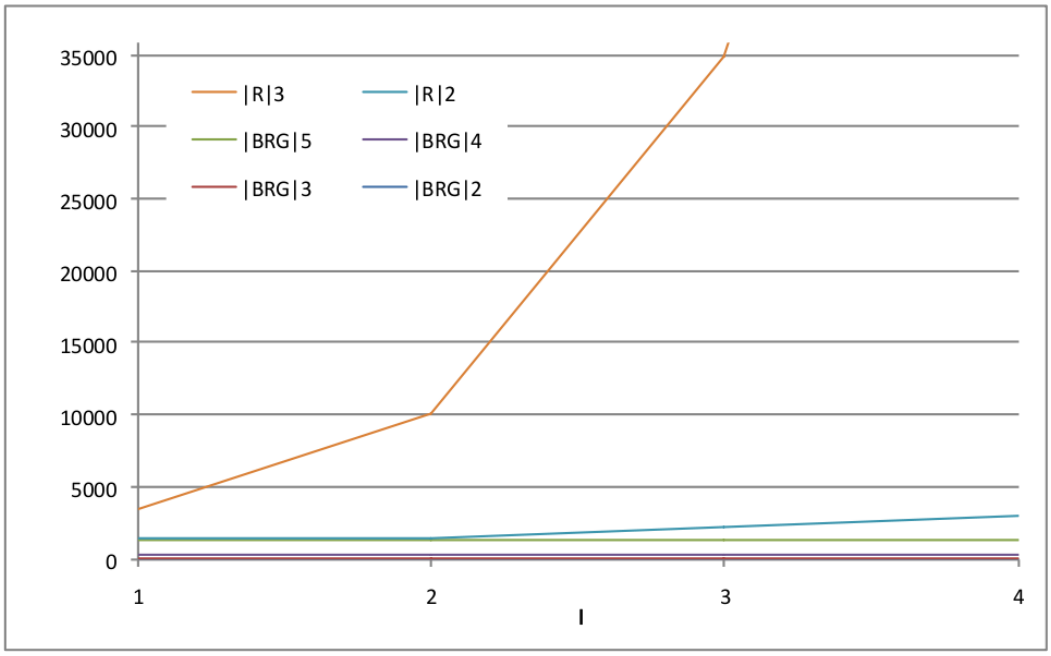

Figura 6.2: Raffronto tra cardinalità del BRG e del grafo di raggiungibilità R in riferimento ai parametri m ed l. per la rete in Figura 6.1.

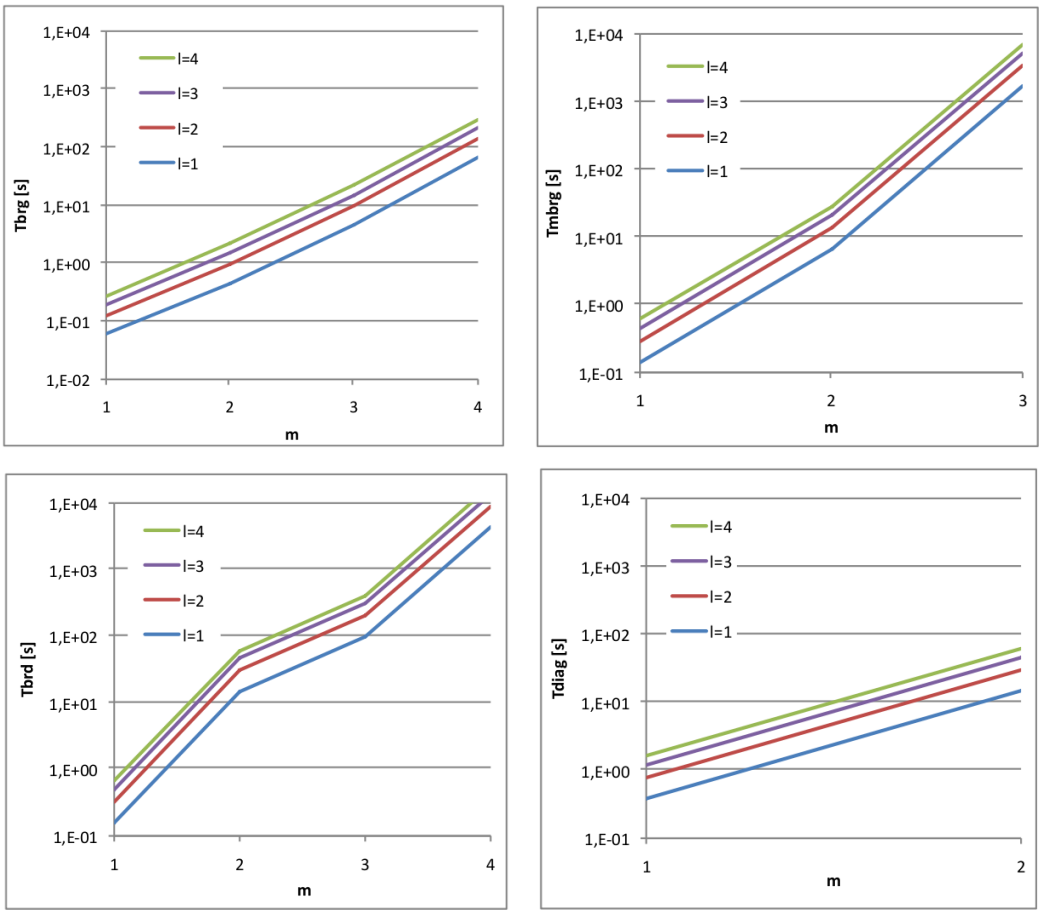

Figura 6.3: Tempi computazionali t_{BRG}, t_{MBRG}, t_{BRD} e t_{Diag} in riferimento ai parametri m ed l per la rete in Figura 6.1 ($d = 0$).

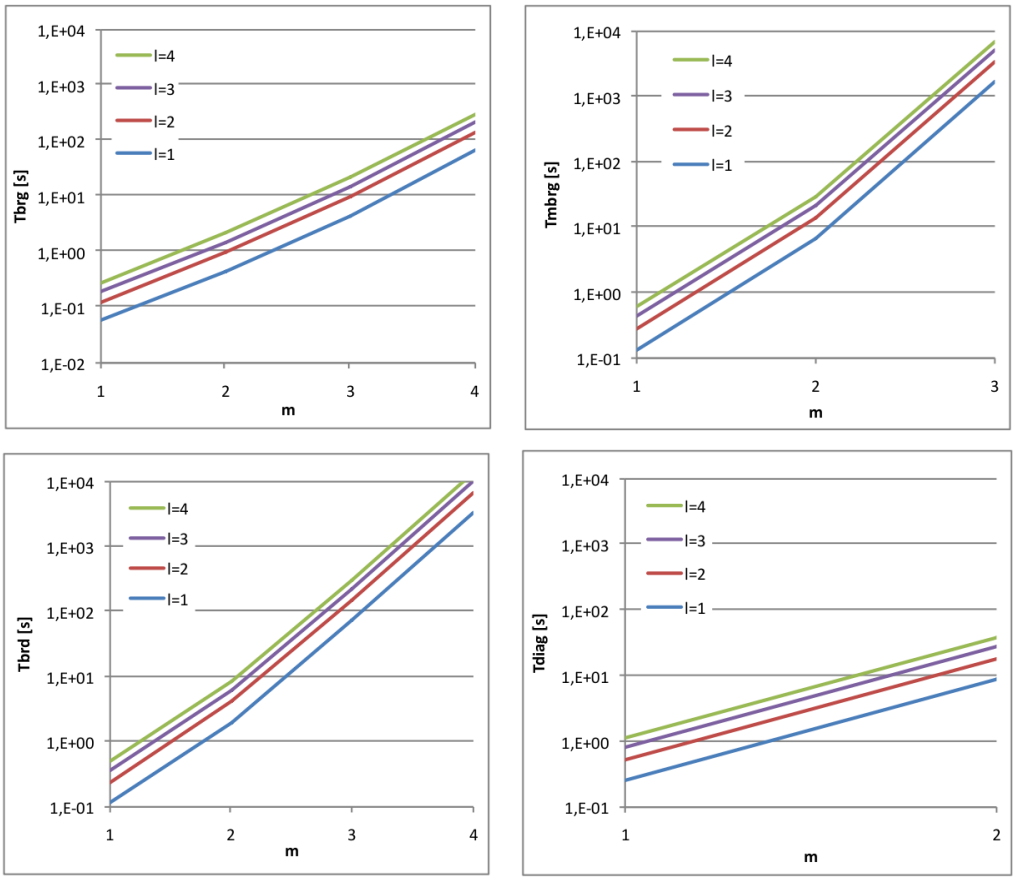

Figura 6.4: Tempi computazionali t_{BRG}, t_{MBRG}, t_{BRD} e t_{Diag} in riferimento ai parametri m ed l per la rete in Figura 6.1 ($d = 1$).

Capitolo 7

Conclusioni e sviluppi futuri

La tesi è dedicata alla diagnosi delle reti di Petri (RdP) etichettate e l'obiettivo principale è proprio quello di fornire uno strumento per la diagnosi e la diagnosticabilità delle stesse.

Il lavoro si basa sugli studi di Cabasino, Giua e Seatzu ([8, 9]) e si identifica nell'implementazione di questi risultati in un toolbox di MATLAB di utilizzo semplice ed intuitivo.

In particolare, è stato analizzato il problema della diagnosi mediante le RdP etichettate, per le quali si è assunto che alcune transizioni della rete siano non osservabili, comprese quelle che identificano dinamiche di guasto. Il loro approccio, applicabile a tutti quei sistemi di rete la cui sottorete non osservabile sia aciclica, è basato sulle nozioni di *Marcatura di base* e di *giustificazione*; la prima identifica l'insieme delle marcature consistenti con una data osservazione, mentre la seconda definisce l'insieme delle transizioni non osservabili il cui scatto è necessario per l'abilitazione della parola stessa. Sono definiti quattro stati di diagnosi, corrispondenti rispettivamente ad un diverso livello di allarme ed è dato un programma, da me scritto, per la determinazione dell'attuale stato di diagnosi per una data osservazione. Cabasino *et al.* hanno dimostrato come per sistemi limitati si possa effettuare off-line gran parte di questa procedura, mediante la costruzione di un grafo, noto come *Grafo di Raggiungibilità di Base* (BRG), di cui ho fornito un programma per la sua definizione.

È stato scritto un programma per per testare la diagnosticabilità di una RdP etichettata, sulla base sul modello presentato in [8], che consta della verifica di condizioni necessarie e sufficienti. Il metodo di Cabasino *et al.* è basato sull'analisi

di due grafi che dipendono dalla struttura della rete e dal modello di guasto. Il primo grafo è detto *Diagnosticatore di Raggiungibilità di Base* (BRD) ed il secondo *Grafo di Raggiungibilità di Base Modificato* (MBRG), di entrambi ho fornito il codice sorgente per la loro definizione.

In futuro, sarebbe interessante effettuare un confronto in termini di complessità computazionale tra il nostro approccio alla diagnosi ed il *Diagnoser approach* di Lafortune. Entrambi, si basano sull'ipotesi di limitatezza della rete. Nel particolare, verrebbero messi a confronto il toolbox realizzato e *DESUMA*, un'integrazione delle librerie UMDES, sviluppato per modellare gli automi a stati finiti e per la verifica di determinate proprietà tra cui quella della diagnosticabilità. In aggiunta, si potrebbe pensare di valutare le metodologie viste in questa tesi per estenderle a tipologie di reti non limitate.

Appendice A

Listati dei programmi

In questo capitolo verranno inseriti i listati di tutti i programmi trattati nel corso del lavoro di tesi

A.1 Costruzione del BRG

```
function [T] =BRG(Pre,Post,M0,F,L,E)

%BRG:  This function builds the basis reachability graph of basis
%      markings of a Petri net.
%
%      ********************************************************************
%                                ## SYNTAX ##
%
%      [T] = BRG(Pre,Post,M0,F,L,E)
%
%      Given a Petri net with its matrices Pre and Post,the initial marking
%      M0, the matrix F, that indicates the Fault class,the matrix L, that
%      indicates the Labeling function, and the Matrix E,that indicates the
%      Alphabet of label, this function returns a matrix T that contains
%      for each row:
%      1) the node's number;
%      2) the basis markings that belong to the node;
%      3) the Xfault vector;
%      4) the transitions enabled by the node;
%      5) the nodes that are reachable with the firing of any transition
%         in association with the j-vector correspondent.
%
%        _____________________________________________________________
%       |   1    |      2       |   3   |        4          |     5      |
%      T=|________|______________|_______|___________________|____________|
%        |# Number|Basis Marking|X fault|Enabled Transitions|Arc-t Matrix|
%        |________|______________|_______|___________________|____________|
%
```

```
%          _____________________________
%         | 6 |       7     |     8      |
%         |___|_____________|____________|
%         |Tag|Enabled label|Arc-l Matrix|
%         |___|_____________|____________|
%
%       F must be a cell array that has as many rows as the number of fault
%       classes, that contains in each row the fault transitions that belong
%       to the corresponding fault class.
%       L must be a cell array that has as many rows as the cardinality of
%       the considered alphabet, that contains in each row the observable
%       transitions having the same label.
%       E must be a cell array that contains in each row a string of
%       characters, each one corresponding to a different label in the
%       considered alphabet. Obviously, the cell array E is ordered
%       according to L.
%
%       See also: MBRG,BRD, showBRG, diagnosability

% Verifica dei Parametri in ingresso

[m,n]=size(Post);
[c,p]=size(M0');
[sizeL,lenghtL]=size(L);
[numTclas,b]= size(F);

if (size(Post)==size(Pre))&(c==1)&(p==m),
elseif size(Post)~=size(Pre)
   fprintf('\n ERROR!! Matrices Pre and Post have different dimension!\n')
   fprintf('\n Put again the data\n')
else
   fprintf('\n  ERROR! dimensions of the  marking are wrong!\n')
end

if (lenghtL==1),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing')
   fprintf('\n  the Labeling Function are wrong!\n')
end

if (b==1),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing')
   fprintf('\n the Fault Class are wrong!\n')
end

if (size(L)==size(E)),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing\n  ')
   fprintf('the Symbol set is not consistent wrt\n  the Labeling function!\n')
end

% Determinazione rispettivamente il numero di transizioni osservabili (no),
% di guasto (nf), regolari(nreg) e non osservabili (nu)

no=0;
for i=1:sizeL
    [sizeLi,lenghtLi]=size(L{i,1});
    no=no+lenghtLi;
end
```

```
nf=0;
for i=1:numTclas
    [sizeFi,lenghtFi]=size(F{i,1});
    nf=nf+lenghtFi;
end

nreg=n-no-nf;

nu=nreg+nf;


% E' strettamente necessario ordinare le transizioni in maniera da avere
% rispettivamente prima quelle osservabili e successivamente le restanti
% non osservabili. Per congruenza con il MBRG faremo in maniera da avere
% le transizioni osservabili, quelle di guasto e per ultime le regolari.

MPre=[];
MPost=[];
MF=[];
ML=[];

% Vettore utilizzato in seguito per tenere traccia dell'originaria
% enumerazione delle transizioni.
% Es: if numbers(i)=j then la transizione tj è stata numerata ti

numbers=zeros(1,n);

% Definizione di un vettore binario tale che
% if(reg(i)==1)
% then la transizione i-esima è una transizione regolare.
% Infine un contatore utile per risettare le matrici definenti funzione di
% etichettatura e classi di guasto

reg=ones(1,n);
count=0;
for i=1:sizeL
    [sizeLi,lenghtLi]=size(L{i,1});
    for j=1:lenghtLi
        count=count+1;
        t=L{i,1}(1,j);
        reg(t)=0;
        numbers(count)=t;
        MPre=[MPre,Pre(:,t)];
        MPost=[MPost,Post(:,t)];
        ML{i,1}(1,j)=count;
    end
end

for i=1:numTclas
    [sizeFi,lenghtFi]=size(F{i,1});
    for j=1:lenghtFi
        count=count+1;
        f=F{i,1}(1,j);
        reg(f)=0;
        numbers(count)=f;
        MPre=[MPre,Pre(:,f)];
        MPost=[MPost,Post(:,f)];
        MF{i,1}(1,j)=count;
    end
end

for i=1:n
```

```
    if (reg(i)==1)
        count=count+1;
        numbers(count)=i;
        MPre=[MPre,Pre(:,i)];
        MPost=[MPost,Post(:,i)];
    end
end


Pre=MPre;
Post=MPost;
L=ML;
F=MF;

% Inizio del conteggio temporale
tic

% Inizializzazione della matrice dati del BRG

T=[{1}{[M0]}{[zeros(1,(numTclas))]}{[zeros(1,no)]}{[(empty_vector(no))]}{[0]}];

c=cell2mat(T(:,6));
d=find(c==0);


% Sinchè esiste almeno un nodo senza TAG, trova le coordinate di tali nodi
% da esplorare e salvale nel vettore d

while ~isempty(d)
d=find(c==0);
%Per tutte le transizioni osservabili:
    for i=1:no

    % Per ciascuna delle marcature di base raggiungibili da Mcurrent con la
    % i-esima transizione Osservabile, a partire dalla prima marcatura non
    % esplorata MM e per ciascuno dei j-vector associati, definisco gli
    % archi del grafo.

        MM=T{d(1,1),2};
        %fprintf('Transizione ', i)
        Mb=Mbasis( Pre , Post ,MM , nu , i);
        %sizeMb21 è il numero delle marcature raggiungibili dal nodo MM
        [sizeMb21,lengthMb21]=size(Mb{2,1});

        for j=1:sizeMb21
            Nodeold=0;
            Mcurrent=Mb{2,1}(j,:);

        % Verifichiamo  se le transizioni di guasto sono abilitate.
        % Questa funzione deve essere ricompilata per essere utilizzata su
        % MAC-OS
            Sol =constrainset(Pre , Post , Mcurrent' , nu , F);

        % Utilizzare questa nel caso in cui non si abbia la possibilità di
        % farlo
        % Sol =faultclass(Pre , Post , Mcurrent' , nu , 0 ,F);

        % Verifichiamo il numero di J-Vector associati alla transizione
        % considerata
            [sizeJi,lenghtJi]=size(Mb{2,2}{j,1});

            for jj=1:sizeJi
```

```
    jcurrent=Mb{2,2}{j,1}(jj,:);
% Ricordando che il riordinamento delle transizioni porta ad
% avere un riordinamento anche del vettore di scatto della
% giustificazione:

    a=[numbers(1,no+1:end);jcurrent];
    temp=sortrows(a')';
    jcurrent=temp(2,:);
% Verifica del fatto che una transizione sia abilitata o meno
    if( ~isempty( find(Mcurrent<0) ) )
        T{d(1,1),4}(1,i)=0;
        continue

% Verifica del fatto che una transizione "cappio" non scatti  a
% meno che il posto corrispondente non sia marcato
    elseif(min(Mcurrent'- Post(:,i))<0)
        T{d(1,1),4}(1,i)=0;
        continue

    else
        %se la transizione è abilitata e porta ad una Mcurrent

        [numofnode, lengthT]=size(T);

    % Verifico che la marcatura non sia già presente nel grafo
        for k=1:numofnode
            if(Mcurrent'==T{k,2})
                T{d(1,1),4}(1,i)=1;
                [sizej,lengthj]=size(T{d(1,1),5}{1,i});
                T{d(1,1),5}{1,i}{sizej+1,1}={[k] [jcurrent]};
                Nodeold=1;
            end
        end

    % Se la Marcatura non è già presente, Nodeold==0, aggiungo
    % un nuovo nodo
        if(Nodeold==0) % creo il nuovo nodo

            [numofnode, lengthT]=size(T);
            T{numofnode+1,1}=[numofnode+1];
            T{numofnode+1,2}=[Mcurrent'];
            T{numofnode+1,3}= Sol;
            A =empty_vector(n-nu);
            T{numofnode+1,4}= zeros(1,no);
            T{numofnode+1,5}= A;
            T{numofnode+1,6}= 0;

        % Aggiorno i valori del nodo considerato

            T{d(1,1),4}(1,i)=1;

        % Questa funzione deve essere ricompilata per essere
        % utilizzata su MAC-OS
            Sol2 =constrainset(Pre , Post , MM , nu , F);
        % Sol2=faultclass(Pre , Post , MM , nu , 0 ,F);

            T{d(1,1),3}=Sol2;
            [sizej,lengthj]=size(T{d(1,1),5}{1,i});
            T{d(1,1),5}{1,i}{sizej+1,1}={[numofnode+1] [jcurrent]};
            T{d(1,1),6}= 0;
            [numofnode, lengthT]=size(T);
        end
```

```
                end
            end
        end
    end

    T{d(1,1),6}= 1;

% Riverifico i nodi non ancora esplorati

    c=cell2mat(T(:,6));
    d=find(c==0);
end

% Inseriamo la nuova matrice degli archi nella colonna numero 7 della
% matrice T del BRG

[sizeT,lenghtT]=size(T);

% Genero l'array in cui inserire il vettore binario delle transizioni
% abilitate

for n=1:sizeT
    T{n,7}=zeros(1,sizeL);
end

% Per tutti i nodi del BRG
for n=1:sizeT
T{n,8}=empty_vector(sizeL);
    %per tutte le etichette
    for i=1:sizeL
[sizeLi,lenghtLi]=size(L{i,1});
%per la j-esima transizione appartenente alla j-esima classe
        count=0;
        for j=1:lenghtLi
            transition=L{i,1}(1,j);
            if((T{n,4}(1,transition))==1)
                T{n,7}(1,i)=1;
                [sizeJi,lenghtJi]=size(T{n,5}{transition});
                for jj=1:sizeJi
                    count=count+1;
                    % {[simbolo][transizione] {[nodo di arrivo][j-vector]}}
                    T{n,8}{count,i}=
                    {[E{i,1}] [numbers(transition)] T{n,5}{1,transition}{jj,1}};
                end
             end
         end
     end
end


toc
t=toc;
end
```

A.2 Costruzione dell'MBRG

```
function [T] =MBRG(Pre,Post,M0,F,L,E)

%MBRG:  This function builds the modified basis reachability graph of
%       basis markings of a Petri net.
%
%       ****************************************************************
%                            ## SYNTAX ##
%
%       [T] = MBRG(Pre,Post,M0,F,L,E)
%
%       Given a Petri net with its matrices Pre and Post, the initial
%       marking M0, the matrix F that indicates the Fault class and the
%       matrix L that  indicates the Labeling function,this function
%       returns a matrix T that contains for each row:
%       1) the node's number;
%       2) the basis markings that belong to the node;
%       3) the Xfault vector;
%       4) the transitions enabled by the node;
%       5) the nodes that are reachable with the firing of any observable
%          or faulty transition in association with the j-vector
%          correspondent.
%          ____________________________________________________________
%         |    1   |      2      |   3    |         4                  |
%       T=|________|_____________|________|____________________________|
%         | # Node |Basis Marking|X fault | Enabled Transitions        |
%         |        |             |        |(observable and faulty)     |
%         |________|_____________|________|____________________________|
%          ____________________________________________
%         |     5      | 6 |      7      |     8      |
%         |____________|___|_____________|____________|
%         |Arc-t Matrix|Tag|Enabled label|Arc-l Matrix|
%         |            |   |             |            |
%         |____________|___|_____________|____________|
%
%      F must be a cell array that has as many rows as the number of fault
%      classes, that contains in each row the fault transitions that belong
%      to the corresponding fault class.
%      L must be a cell array that has as many rows as the cardinality of
%      the considered alphabet, that contains in each row the observable
%      transitions having the same label.
%      E must be a cell array that contains in each row a string of
%      characters, each one corresponding to a different label in the
%      considered alphabet. Obviously, the cell array E is ordered
%      according to L.
%
%       See also: BRG, BRD, showMBRG, diagnosability


% Verifica dei Parametri in ingresso

[m,n]=size(Post);
[c,p]=size(M0');
[sizeL,lenghtL]=size(L);
[numTclas,b]= size(F);

if (size(Post)==size(Pre))&(c==1)&(p==m),
elseif size(Post)~=size(Pre)
   fprintf('\n ERROR!! Matrices Pre and Post have different dimension!\n')
```

```
   fprintf('\n Put again the data\n')
else
   fprintf('\n  ERROR! dimensions of the  marking are wrong!\n')
end

if (lenghtL==1),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing')
   fprintf('\n  the Labeling Function are wrong!\n')
end

if (b==1),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing')
   fprintf('\n the Fault Class are wrong!\n')
end

if (size(L)==size(E)),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing')
   fprintf('\n  the Symbol set is not consistent wrt')
   fprintf('\n  the Labeling function!\n')
end

% Determino rispettivamente il numero di transizioni osservabili (no), di
% guasto (nf), regolari(nreg) e non osservabili (nu)

no=0;
for i=1:sizeL
    [sizeLi,lenghtLi]=size(L{i,1});
    no=no+lenghtLi;
end


nf=0;
for i=1:numTclas
    [sizeFi,lenghtFi]=size(F{i,1});
    nf=nf+lenghtFi;
end

nreg=n-no-nf;

nu=nreg+nf;

% E' strettamente necessario ordinare le transizioni in maniera da avere
% rispettivamente prima quelle osservabili, poi quelle di guasto e,
% successivamente, le restanti regolari.

MPre=[];
MPost=[];
MF=[];
ML=[];

% Vettore utilizzato in seguito per tenere traccia dell'originaria
% enumerazione delle transizioni.
% Es: if numbers(i)=j then la transizione tj è stata numerata ti

numbers=zeros(1,n);

% Definizione di un vettore binario tale che
% if(reg(i)==1)
% then la transizione i-esima è una transizione regolare.
```

```
% Infine un contatore utile per risettare le matrici definenti funzione di
% etichettatura e classi di guasto

reg=ones(1,n);
count=0;
for i=1:sizeL
    [sizeLi,lenghtLi]=size(L{i,1});
    for j=1:lenghtLi
        count=count+1;
        t=L{i,1}(1,j);
        reg(t)=0;
        numbers(count)=t;
        MPre=[MPre,Pre(:,t)];
        MPost=[MPost,Post(:,t)];
        ML{i,1}(1,j)=count;
    end
end

for i=1:numTclas
    [sizeFi,lenghtFi]=size(F{i,1});
    for j=1:lenghtFi
        count=count+1;
        f=F{i,1}(1,j);
        reg(f)=0;
        numbers(count)=f;
        MPre=[MPre,Pre(:,f)];
        MPost=[MPost,Post(:,f)];
        MF{i,1}(1,j)=count;
    end
end

for i=1:n
    if (reg(i)==1)
        count=count+1;
        numbers(count)=i;
        MPre=[MPre,Pre(:,i)];
        MPost=[MPost,Post(:,i)];
    end
end

Pre=MPre;
Post=MPost;
L=ML;
F=MF;

% Inizio del conteggio temporale
tic

% Inizializzazione della matrice dati del MBRG

T=[{1}{[M0]}{[zeros(1,(numTclas))]}{[zeros(1,(n-nreg))]}
                                        {[(empty_vector(n-nreg))]}{[0]}];

c=cell2mat(T(:,6));
d=find(c==0);

% Sinchè esiste almeno un nodo senza TAG, trova le coordinate di tali nodi
% da esplorare e salvale nel vettore d


while ~isempty(d)
       d=find(c==0);
```

```
for i=1:n-nreg

    % Per ciascuna delle marcature di base raggiungibili Mcurrent con la
    % i-esima transizione Osservabile o di guasto a partire dalla prima
    % marcatura non esplorata MM e per ciascuno dei j-vector associati,
    % definisco gli archi del grafo.

    MM=T{d(1,1),2};
    Mb=Mbasis( Pre , Post ,MM , nreg , i );
    %sizeMb21 è pari al numero delle marcature raggiungibili dal nodo MM
    [sizeMb21,lengthMb21]=size(Mb{2,1});

    %per tutte le marcature raggiungibili e il j-vector corrispondente
    for j=1:sizeMb21

        Nodeold=0;
        Mcurrent=Mb{2,1}(j,:);

        % Verifichiamo  se le transizioni di guasto sono abilitate
        % Questa funzione deve essere ricompilata per essere utilizzata su
        % MAC-OS
        Sol =constrainset(Pre , Post , Mcurrent' , nu , F);
        %Sol=faultclass(Pre , Post , Mcurrent' , nu, 0 ,F);

        % Verifichiamo il numero di J-Vector associati alla transizione
        % considerata
        [sizeJi,lenghtJi]=size(Mb{2,2}{j,1});

        for jj=1:sizeJi
            jcurrent=Mb{2,2}{j,1}(jj,:);

            % Verifica del fatto che una transizione sia abilitata o meno
            if( ~isempty( find(Mcurrent<0) ) )
                T{d(1,1),4}(1,i)=0;
                continue

            % Verifica del fatto che una transizione "cappio" non scatti
            % a meno che il posto corrispondente non sia marcato
            elseif(min(Mcurrent'- Post(:,i))<0)
                T{d(1,1),4}(1,i)=0;
                continue

            else %se la transizione è abilitata e porta ad una Mcurrent
                [numofnode, lengthT]=size(T);

                % Verifico che la marcatura non sia già presente nel grafo
                for k=1:numofnode
                    if(Mcurrent'==T{k,2})
                        T{d(1,1),4}(1,i)=1;
                        [sizej,lengthj]=size(T{d(1,1),5}{1,i});
                        T{d(1,1),5}{1,i}{sizej+1,1}={[k] [jcurrent]};
                        Nodeold=1;
                    else
                    end
                end

                % Se la Marcatura non è già presente, aggiungo un nuovo
                % nodo
                if(Nodeold==0) % creo il nuovo nodo

                    [numofnode, lengthT]=size(T);
```

```
                    T{numofnode+1,1}=[numofnode+1];
                    T{numofnode+1,2}=[Mcurrent'];
                    T{numofnode+1,3}= Sol;
                    T{numofnode+1,4}= zeros(1,n-nreg);
                    A =empty_vector(n-nreg);
                    T{numofnode+1,5}= A;
                    T{numofnode+1,6}= 0;

                    % Aggiorno i valori del nodo considerato

                    T{d(1,1),4}(1,i)=1;

                    % Questa funzione deve essere ricompilata per essere
                    % utilizzata su MAC-OS
                    Sol2 =constrainset(Pre , Post , MM , nu , F);
                    % Sol2=faultclass(Pre , Post , MM , nu, 0 ,F);

                    T{d(1,1),3}=Sol2;

                    [sizej,lengthj]=size(T{d(1,1),5}{1,i});
                    T{d(1,1),5}{1,i}{sizej+1,1}={[numofnode+1] [jcurrent]};
                    T{d(1,1),6}= 0;


                    [numofnode, lengthT]=size(T);
                end
            end
        end
    end
end
T{d(1,1),6}= 1;

% Riverifico i nodi non ancora esplorati
c=cell2mat(T(:,6));
d=find(c==0);
end

% Inseriamo la nuova matrice degli archi nella colonna numero 7 della
% matrice T dell'MBRG

[sizeT,lenghtT]=size(T);
nArc=sizeL+numTclas;

% Genero l'array in cui inserire il vettore binario delle transizioni
% abilitate

for n=1:sizeT
    T{n,7}=zeros(1,nArc);
end
% Per tutti i nodi dell'MBRG
for n=1:sizeT
T{n,8}=empty_vector(nArc);
    % per tutte le etichette
    for i=1:sizeL
[sizeLi,lenghtLi]=size(L{i,1});
%per la j-esima transizione appartenente alla j-esima classe
        count=0;
        for j=1:lenghtLi
            transition=L{i,1}(1,j);
            if((T{n,4}(1,transition))==1)
                T{n,7}(1,i)=1;
                [sizeJi,lenghtJi]=size(T{n,5}{transition});
```

```
                for jj=1:sizeJi
                    count=count+1;
                    % {[simbolo][transizione] {[nodo di arrivo][j-vector]}}
                    T{n,8}{count,i}=
                    {[E{i,1}] [numbers(transition)] T{n,5}{1,transition}{jj,1}};
                end
            end
        end
    end

    % Potrebbe darsi che non ci siano transizioni di guasto, dunque
    % effettuiamo la prima verifica
    [si,le]=size(F);

    if(si~=0)
       for k=1:numTclas
        %F{k,1} è un vettore riga contenente le transizioni della classe
        %k-esima di guasto
        [sizeFk,lenghtFk]=size(F{k,1});
        countf=0;
        %per la p-esima transizione appartenente alla k-esima classe di
        %guasto
        for p=1:lenghtFk
            %leggi il numero della transizione di guasto
            faulty=F{k,1}(1,p);
            %se tale transizione è abilitata, ossia se esiste un arco
            %uscente dal nodo
            if((T{n,4}(1,faulty))==1)
                T{n,7}(1,sizeL+k)=1;
                [sizeJi,lenghtJi]=size(T{n,5}{faulty});
                for jj=1:sizeJi
                    countf=countf+1;
                    % {[no etichetta][t_failure] {[nodo di arrivo][j-vector]}}
                    T{n,8}{countf,sizeL+k}=
                            {0,[numbers(faulty)] T{n,5}{1,faulty}{jj,1}};
                end
            end
        end
    end
end

end


toc
t=toc;
end
```

A.3 Costruzione del BRD

```
function [D] =BRD(T,Pre,Post,M0,F,L,E)

%  BRD:   This function builds the basis reachability diagnoser of basis
%         markings of a Petri net.
%
%         ************************************************************
%                                ## SYNTAX ##
%
%         [D] = BRD(T,Pre,Post,M0,F,L,E)
%
%         Given a Petri net with its Basis Reachability Graph (T), matrices
%         Pre and Post, the initial marking M0, the matrix F that indicates
%         the Fault class and the matrix L that indicates the Labeling
%         function, this function returns a matrix D that contains for each
%         row:
%         1) the node's number;
%         2) the basis markings that belong to the node;
%         3) the Xfault vector;
%         4) the transitions enabled at the node;
%         5) the Diagnosis State;
%         6) the couple of the enabled transition  and the correspondent
%            destination's node.
%
%         NB:D{1,2}(1,1) is used to keep memory of the number of net's
%            place
%
%                 ____________________________________________________
%           |      1      |              2                  |    3     |
%           |_____________|_________________________________|__________|
%         D=|Node's Number| [Basis Marking,Xfault,h?{N,F}]| Diagnosis|
%           |             |                                 |   State  |
%           |_____________|_________________________________|__________|
%                  ____________________________
%           |         4         |   5   | 6 |
%           |___________________|_______|___|
%           |   Enabled labels  |Marking|Tag|
%           | & destination node| index |   |
%           |___________________|_______|___|
%
%         F must be a cell array that has as many rows as the number of
%         fault classes, that contains in each row the fault transitions
%         that belong to the corresponding fault class.
%         L must be a cell array that has as many rows as the cardinality
%         of the considered alphabet, that contains in each row the
%         observable transitions having the same label.
%         E must be a cell array that contains in each row a string of
%         characters, each one corresponding to a different label in the
%         considered alphabet. Obviously, the cell array E is ordered
%         according to L.
%
%         See also: MBRG, BRG,showBRD, diagnosability


[m,n]=size(Post);
[c,p]=size(M0');
[numTclas,b]= size(F);
[sizeL,lenghtL]= size(L);
```

```
% Verifica dei parametri in ingresso

if (size(Post)==size(Pre))&(c==1)&(p==m),
elseif size(Post)~=size(Pre)
   fprintf('\n ERROR!! Matrices Pre and Post have different dimension!\n')
   fprintf('\n Put again the data\n')
else
   fprintf('\n  ERROR! dimensions of the  marking are wrong!\n')
end

if (lenghtL==1),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing')
   fprintf('\n  the Labeling Function are wrong!\n')
end

if (b==1),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing')
   fprintf('\n the Fault Class are wrong!\n')
end

if (size(L)==size(E)),
else
   fprintf('\n  ERROR! dimensions of the Matrix representing\n  ')
   fprintf('the Symbol set is not consistent wrt\n  the Labeling function!\n')
end



% Determino rispettivamente il numero di transizioni osservabili (no), di
% guasto (nf), regolari(nreg) e non osservabili (nu)

no=0;
for i=1:sizeL
    [sizeLi,lenghtLi]=size(L{i,1});
    no=no+lenghtLi;
end


nf=0;
for i=1:numTclas
    [sizeFi,lenghtFi]=size(F{i,1});
    nf=nf+lenghtFi;
end

nreg=n-no-nf;
nu=n-no;

numbers=zeros(1,n);
reg=ones(1,n);
c=0;

for i=1:sizeL
    [sizeLi,lenghtLi]=size(L{i,1});
    for j=1:lenghtLi
        c=c+1;
        t=L{i,1}(1,j);
        reg(t)=0;
        numbers(c)=t;
    end
end
```

```
for i=1:numTclas
    [sizeFi,lenghtFi]=size(F{i,1});
    for j=1:lenghtFi
        c=c+1;
        f=F{i,1}(1,j);
        reg(f)=0;
        numbers(c)=f;
    end
end

for i=1:n
    if (reg(i)==1)
        c=c+1;
        numbers(c)=i;
    end
end

% La Verifica dei Parametri in ingresso è effettuata all'interno della
% chiamata alla funzione BRG

% Starting of timing computation of Basis Reachability Diagnoser
tic


[m,n]=size(Post);
[c,p]=size(M0');
[numTclas,b]= size(F);
[sizeL,lenghtL]= size(L);


% Initialization of the Basis Reachability Diagnoser

% Facciamo attenzione al fatto che h(i)=N èindicato come "-1" mentre h(i)=F
% è indicato come "-2", per poter lavorare con vettori unicamente numerici.
% (Credo non si possano avere vettori misti, ossia alfanumerici, dunque
% sarebbe necessario separare le due informazioni)

D=[{1}, {zeros(1,p+2*numTclas)},{[]},{[]},{[]} {0}];

% La prima riga della seconda colonna è utilizzata per salvare il numero di
% posti della rete, parametro utilizzato nella funzione showBRD
D{1,2}(1,1)=-p;

D{1,2}(2,1:p)=M0';
D{1,2}(2,p+1:p+numTclas)=T{1,3};
D{1,2}(2,p+1+numTclas:p+2*numTclas)=-1*ones(1,numTclas);
D{1,3}=T{1,3};
D{1,4}=empty_vector(sizeL);
D{1,5}=[0;0];

% While nodes with no Tag Exist we select one among them and explore it
% [ Remember that a node not already explored has a null tag ]

c=cell2mat(D(:,6));
% We create "d" as a vector with the coordinates of the nodes not explored
d=find(c==0);

while ~isempty(d)
    d=find(c==0);
    % scelto un nodo del BRD da esplorare, verifico per tutte le marcature
    % che gli appartengono l'insieme delle marcature raggiungibili per
```

```
% ciascuna delle etichette possibili.
for z=1:sizeL
    % Definisco un puntatore per la scrittura. (Se count==3 allora ci
    % sono stati già memorizzati tre archi etichettati 'z' dal nodo del
    % BRD in considerazione.
    clear Dnew Dold
    count=0;
    [numMbasisD,b]= size(D{d(1,1),2});
    Dnew=[{[]}, {[]},{[]},{[]},{[0]}, {0}];
    Dnew{1,2}=zeros(1,p+2*numTclas);
    Dnew{1,4}=empty_vector(sizeL);
    % Per tutte le marcature di Base appartenenti al nodo del BRD in
    % analisi:
    for m=1:numMbasisD
        if(D{d(1,1),2}(m,1)<0)
            continue
        end
        Mcurrent=D{d(1,1),2}(m,1:p);
        Hcurrent=D{d(1,1),2}(m,p+numTclas+1:p+2*numTclas);
        [numofnodeT, lengthT]=size(T);

        % Cerco la Marcatura in analisi sul BRG
        for j=1:numofnodeT
            if(Mcurrent==(T{j,2})')
                % Trovata la Marcatura sul BRG salvo le coordinate del
                % nodo e la cardinalità massima degli archi associati
                % ad una etichetta abilitata
                [numofarcMax,numoflabel]= size(T{j,8});
                currentNode=j;
            end
        end

        if (T{currentNode,7}(1,z)==1)
            % Se l'etichetta z-esima è abilitata, aggiungi la marcatura
            % raggiunta al nuovo nodo con i parametri corrispondenti,
            % altrimenti passa alla marcatura successiva

            for i=1:numofarcMax
                if(size(T{currentNode,8}{i,z})~=[0,0])
                    count=count+1;
                    nextNode=T{currentNode,8}{i,z}{1,3}{1,1};
                    % RIcordando che la giustificazione deve tenere
                    % conto del riordinamento delle transizioni:
                    jcurrent=T{currentNode,8}{i,z}{1,3}{1,2};
                    MbNext=(T{nextNode,2})';

                    %vettore utile in seguito per la verifica di scatti
                    %di guasto
                    temp=[no+1:n;jcurrent];

                    % Definizione parametri del nuovo nodo

                    % Indice della Marcatura
                    Dnew{1,5}(count,1)=nextNode-1;

                    % Marcatura di base in seconda colonna
                    Dnew{1,2}(count,1:p)=MbNext;

                    % X-fault corrispondente in terza colonna
                     Dnew{1,2}(count,p+1:p+numTclas)=T{nextNode,3};
                    % Definisco un un vettore binario (hitF),associato
                    % al percorso, che tenga conto della presenza di
```

```
                    % transizioni di guasto
                    hitF=zeros(numTclas);
                    for f=1:numTclas
                        [sizeFi,lenghtFi]= size(F{f,1});

                        % Definisco hitFi come vettore di
                        % scatto relativo alle sole transizioni
                        % della f-esima classe di guasto
                        hitFi=0;
                        for ff=1:lenghtFi
                            tf=F{f,1}(1,ff);
                            findtf=find(temp(1,:)==tf);
                            size(findtf);
                            if(size(findtf)==[1,1])
                                hitFi(ff)=jcurrent(findtf);
                            end
                        end
                        % Definisco hitF come vettore binario tale
                        % che hitF(f)=1, se un guasto della f-esima
                        % classe scatta nel percorso Mcurrent->MbNext,
                        % e hitF(f)=0, se viceversa.

                        if (any(hitFi))
                           hitF(f)=1;
                        else
                            hitF(f)=0;
                        end

                        % Definizione di h(i) \in {N,F}|{-1,-2}
                        if (Hcurrent(f)== -2)
                            Dnew{1,2}(count,p+numTclas+f)= -2;
                        else
                            if (hitF(f)==0)
                                Dnew{1,2}(count,p+numTclas+f)= -1;
                            else
                             Dnew{1,2}(count,p+numTclas+f)= -2;
                            end
                        end
                    end
                % Se non è presente un arco attivo, passa alla
                % transizione successiva associata alla stessa
                % etichetta.
                else
                    continue
                end
            end
        % Se dalla Mbcurrent non è abilitata l'etichetta z-esima, passa
        % alla marcatura di base successiva
        else
            continue
        end
    end


% Se la riga relativa alla marcatura è nulla, ossia se rimane
% settata al valore a cui è stata inizializzata, passa alla
% iterazione successiva e cerca un nodo non ancora esplorato

if(Dnew{1,2}==0)
    continue
end
```

```
% La costruzione del BASIS REACHABILITY DIAGNOSER può portare ad
% avere nello  stesso nodo la medesima marcatura con il medesimo
% valore di "h={N,F}", un'eventualità che non ci interessa.
% Eliminiamo, dunque, le ridondanze:

A=Dnew{1,2};
[a,b]=size(Dnew{1,2});
for bb=1:a
    for aa=1:a
        if (aa~=bb &(Dnew{1,2}(bb,:)==A(aa,:)))
            A(aa,:)=-10;
            Dnew{1,2}(aa,:)=-10;
        end
    end
end

B=[];
clear A
countM=0;
for bb=1:a
    if (Dnew{1,2}(bb,:)~=-10)
        countM=countM+1;
        A(countM,:)=Dnew{1,2}(bb,:);
        B(countM,1)=Dnew{1,5}(bb,1);
    end
end
Dnew{1,2}=A;
Dnew{1,5}=B;

% Stato del diagnosticatore in terza colonna

H=Dnew{1,2}(:,p+numTclas+1:p+2*numTclas);
Fault=Dnew{1,2}(:,p+1:p+numTclas);

for f=1:numTclas
    if(Fault(:,f)==0 & H(:,f)==-1)
        Dnew{1,3}{1}(1,f)=0;
    elseif(H(:,f)==-2)
        Dnew{1,3}{1}(1,f)=3;
        % Almeno un elemento di X-fault deve essere unitario
    elseif(sum(Fault(:,f))>0 & H(:,f)==-1)
        Dnew{1,3}{1}(1,f)=1;
    else
        Dnew{1,3}{1}(1,f)=2;

    end
end

% Terminata la definizione del nodo, non ci resta che verificare
% che non sia già presente nel BRD
% Memorizziamo tutte le colonne, fatta eccezione per numero di nodo
% e TAG.

DDnew=Dnew{1,2};
DDold=[];
% Determino il numero di nodi del BRD (numofnodeD)
[numofnodeD,lengthD]=size(D);
% Determino la cardinalità del nodo in analisi(sizeDDnew)
[sizeDDnew,lenghtDDnew]=size(DDnew);
exist=0;
for dd=1:numofnodeD
    DDold=D{dd,2};
```

```
            [sizeDDold,lenghtDDold]=size(DDold);

            % Se la cardinalità dei nodi è la stessa, allora verifico che
            % siano rappresentativi degli stessi parametri (M & h).

            % NB: In realtà, avendo la colonna degli indici delle
            % Marcature, sarebbe + snello ed intuitivo, verificato che la
            % cardinalità sia la stessa, ordinare il vettore colonna e
            % verificare che sia lo stesso

            if (sizeDDold==sizeDDnew)
                for ddd=1:sizeDDnew
                    for dddd=1:sizeDDold
                        if (DDnew(ddd,:)==DDold(dddd,:))
                            DDold(dddd,:)=-10;
                        break
                        end
                    end
                end

                if(DDold==-10)
                    D{d(1,1),4}{z}{2}=dd;
                    D{d(1,1),4}{z}{1}=E{z,1};
                    exist=1;
                    break;
                end
            end
        end
        if(exist==1)
            continue
        else
            D{numofnodeD+1,1}=numofnodeD+1;
            D{numofnodeD+1,2}=Dnew{1,2};
            D{numofnodeD+1,3}=Dnew{1,3}{1};
            D{numofnodeD+1,5}=Dnew{1,5};
            D{numofnodeD+1,6}=0;
            D{d(1,1),4}{z}{2}=numofnodeD+1;
            D{d(1,1),4}{z}{1}=E{z,1};

        end

        % Se il nodo non è ancora presente, lo aggiungo a termine del BRD,
        % setto tag=1 e segnalo che l'arco del nodo precedente arriva nel
        % nuovo nodo.

    end
    % Riverifico i nodi non ancora esplorati
    D{d(1,1),6}=1;
    c=cell2mat(D(:,6));
    d=find(c==0);
end
toc
t=toc;
end
```

A.4 Verifica della diagnosticabilità

```
function diagnosability(MBRG,BRD)

%DIAGNOSABILITY: This function verifies necessary and sufficient conditions
%                for diagnosability of bounded Petri nets system.
%                The method to test diagnosability is based on the analysis
%                of two graphs, the MBRG (Modified Basis Reachability
%                Graph) and the BRD (Basis Reachability Diagnoser).
%
%                NB: as soon as we find an indeterminate cycle for a fault
%                class, we stop to search other indeterminate cycle fort
%                and this class conclude that the system is notdiagnosable
%                wrt that fault class.
%
%                *********************************************************
%                                    ## SYNTAX ##
%
%                diagnosability(MBRG,BRD)
%
%                Given the MBRG and the associated BRD, this function
%                returns faulty class that are not diagnosable.
%
%                See also: MBRG, BRG, BRD, showMBRG, showBRG, showBRD


% Verifica del fatto che il numero di argomenti in ingresso alla funzione
% sia esattamente due. In caso Contrario viene plottato un messaggio di
% errore

ni=nargin;
error(nargchk(2,2,ni));

% Inizio del calcolo del tempo impegato per la verifica delle condizioni
% necessarie e sufficienti per la diagnosticabilità.

tic

% Inizializzazione dei vettore 'N', dove andare a scrivere la sequenza dei
% nodi attraversati lungo il percorso. Il vettore 'W', dove andremo a
% scrivere la parola osservata consistente al percorso eseguito, verrà
% inizializzato nel ciclo successivo.

N=1;
diag=ones(size(BRD{1,3}));
CYCLE=[];
WORD=[];
%W='a';
% Contatore cicli

count_c=0;

% Sinchè esiste almeno un nodo non morto o non appartenente ad un ciclo,
% esso apparterrà alla matrice/vettore N e verrà, così, esplorato.

[d,dd]=size(N);

while d~=0
    % I percorsi non terminati in un ciclo o in un nodo "morto" vanno ad
    % essere esplorati.
```

```
    [d,dd]=size(N);
    statononvoluto =zeros(1,d);
cl=ones(1,d);
    % Contatore degli archi uscenti aggiuntivi ai nodi esplorati
count=0;
    for i=1:d
        n_current=N(i,dd);
        [one,numOfArc]=size(BRD{n_current,4});
        x=0;
        for k=1:numOfArc
            if(size(BRD{n_current,4}{k})~=0)
                n_next=BRD{n_current,4}{k}{2};
                label =BRD{n_current,4}{k}{1};
                % Aggiungo il nodo appena raggiunto del vettore contenente
                % i nodi componenti il percorso e l'etichetta nel vettore
                % contenente la parola consistente
                if(x==0)
                    N(i,dd+1)=n_next;
                    W(i,dd)=label;
                    x=1;
                    cl(1,i)=0;

                    if BRD{n_next,3}==3
                        statononvoluto(1,i) =1;
                    end

                else
                    count=count+1;
                    N(d+count,:)=[N(i,1:dd),n_next];
                    W(d+count,:)=[W(i,1:dd-1),label];
                    cl(1,d+count)=0;

                    if BRD{n_next,3}==3
                        statononvoluto(1,d+count) =1;
                    else
                        statononvoluto(1,d+count) =0;
                    end

                end
            end
        end
    end


    [d,dd]=size(N);

    % cancello i percorsi terminanti in nodi da cui non é abilitata
% alcuna transizione
for i=d:-1:1
        if cl(1,i)==1
            N(i,:)=[];
            W(i,:)=[];
            statononvoluto(1,i)=[];
        end
    end

    [d,dd]=size(N);

    % cancello i percorsi terminanti in nodi il cui stato di diagnosi è
    % pari a 3, poichè non di interesse.
for i=d:-1:1
        if statononvoluto(1,i)==1
```

```
            N(i,:)=[];
            W(i,:)=[];
        end
    end


    [d,dd]=size(N);

    % Verifica della presenza di eventuali cicli tra i percorsi appena
    % determinati.

    % Si faccia attenzione alla struttura di
    %       _____________________________________
    %      |     1     |     2     |    3      |
    %      |___________|___________|___________|
    % WORD=| Proiezione| Proiezione|  binary   |
    %      |Osservabile|Osservabile|  vector   |
    %      |  Cammino  |   Ciclo   |    b      |
    %      |___________|___________|___________|
    %
    % Dove binary-vector è tale che:
    %                          _
    %                         |  1 se il ciclo è candidato ad essere
    %                         |    indeterminato per la i-esima classe di
    %                b(i)= <     guasto
    %                         |
    %                         |_ 0 altrimenti
%       _____________________________
    %       |     1      |     2      |
    %       |____________|____________|
    % CYCLE=|  Sequenza  |  Sequenza  |
    %       |    Nodi    |    Nodi    |
    %       |Attraversati|Attraversati|
    %       |nel  cammino| nel  ciclo |
    %       |____________|____________|

    count_c=0;

    for i=d:-1:1
        n_current=N(i,dd);
        exist=find(N(i,1:dd-1)==n_current);
        [e,ee]=size(exist);
        if(ee~=0)
            count_c=count_c+1;
            CYCLE{count_c,1}=N(i,1:exist-1);
            CYCLE{count_c,2}=N(i,exist:dd);
            WORD{count_c,1}=W(i,1:exist-1);
            WORD{count_c,2}=W(i,exist:dd-1);
            WORD{count_c,3}=zeros(size(BRD{1,3}));

            % Se il percorso é identificato come ciclo, cessa di essere
            % esplorato (ciclo elementare)
            N(i,:)=[];
            W(i,:)=[];
        end
    end
    %*********************************************************************
    %   Si vuole fare in maniera che qualora si venga a determinare un
    %   ciclo candidato ad essere indeterminato, esso debba esser esplorato
    %   nell'MBRG per la verifica completa, al fine di non esplorare tutti
    %   i cicli SE NON NECESSARIO.
    %*********************************************************************
```

```
    [sizeCYCLE,lenghtCYCLE]=size(CYCLE);
    [b,numTclas]= size(BRD{1,3});

    % Se tra i percorsi analizzati vengono trovati dei cicli
    if sizeCYCLE~=0
        % viene verificato che i cicli trovati siano indeterminati

        % Dunque, non ci resta che verificare che siano di soli nodi
        % con stato di diagnosi appartenente a {1,2}
        for i=sizeCYCLE:-1:1
            for k=1:numTclas
                [sizeCYCLEi,lenghtCYCLEi]=size(CYCLE{i,2});
                for j=1:lenghtCYCLEi
                    node=CYCLE{i,2}(1,j);
                    diag_state=BRD{node,3}(k);
                    if (diag_state==0)
                        WORD{i,3}(k)=0;
                        break
                    elseif(diag_state==3)
                        WORD{i,3}(k)=0;
                        break
                    else
                        WORD{i,3}(k)=1;
                    end
                end
            end

            if(WORD{i,3}==0)
                WORD(i,:)=[];
                CYCLE(i,:)=[];
            end
        end
    end

% sizeCYCLE= numero di cicli candidati ad essere indeterminati
    [sizeCYCLE,lenghtCYCLE]=size(CYCLE);
    [b,numTclas]= size(BRD{1,3});

if sizeCYCLE~=0

        % A questo punto non ci resta che stabilire se i cicli siano
        % indeterminati o meno.
        % Per ciascuna classe di guasto viene verificato che esista almeno
        % un ciclo nel BRD tale che ce ne sia una coppia nel MBRG con
        % medesima proiezione osservabile, che nel primo sia presente il
        % guasto mentre nel secondo no.

        % La matrice/vettore
        candidates=cell2mat(WORD(:,3));

        % Verifico il numero di cicli candidati ad essere indeterminati
        [numcycle,numTclas]=size(candidates);

        if numcycle==0
            continue
        end


        % Per ciascuna classe di guasto devo verificare che esista un ciclo
        % indeterminato (nell'MBRG)
        PATH{1,1}=1;
```

```
% Indica se il guasto si sia verificato o no, inizialmente é tutto
% settato a False
PATH{1,2}=zeros(1,numTclas);

    %************************************************************
    %   E' Necessario inserire tutti quei nodi raggiungibili con
    %   transizioni silenziose a partire dal nodo iniziale
    %************************************************************

[NN,hit] =faultreachability(MBRG,1);
[sizeNN,lenghtNN]=size(NN);
countf=1;
if(lenghtNN>1)



    for iii=1:sizeNN
        for jjj=2:lenghtNN
            current_node=NN(iii,jjj);
            % Se il nodo raggiunto non é ancora presente tra quelli
            % raggiungibili dal nodo iniziale, creo un nuovo
            % percorso
            founded=find(PATH{1,1}==current_node);
            [sizefounded,lenghtfounded]=size(founded);

            if sizefounded==0
                countf=countf+1;
                PATH{1,1}(countf,1)=current_node;
                PATH{1,2}(countf,:)=hit{iii,jjj-1};
            else
                % Se é presente ma é raggiunto con una differente
                % dinamica di guasto, creo un nuovo percorso
                for iiii=1:sizefounded
                    if hit{iii,jjj-1}~=PATH{1,2}(founded(iiii),:)
                        countf=countf+1;
                        PATH{1,1}(countf,1)=current_node;
                        PATH{1,2}(countf,:)=hit{iii,jjj-1};
                    end
                end
            end
        end
    end
end
    %***********************************
    %   Transizioni di guasto verificate
    %***********************************

PATHold=PATH;

% Per ogni classe di guasto
for c=1:numTclas

    % Memorizzo gli indici relativi ai cicli che potrebbero essere
    % indeterminati per la classe di guasto c-esima
    candidatesc=find(candidates(:,c)==1);
    [sizeCand,lenghtCand]=size(candidatesc);

    % Per tutti quei cicli del BRD candidati ad essere
    % indeterminati per quella classe di guasto c-esima
    for cc=1:sizeCand
        % Se la classe non è diagnosticabile, non ci interessa
```

```
            % determinarne un secondo ciclo indeterminato!!!
            if(diag(c)==0)
                continue
            else
            PATH=PATHold;

            % Per ogni ciclo trovato,
            % leggo la parola che andremo a verificare sull'MBRG
            cycle=WORD{candidatesc(cc),2};
            path=WORD{candidatesc(cc),1};
            [one,leghtc]=size(cycle);
            [one,leghtp]=size(path);

        % Verifico che tale parola corrisponda ad almeno una coppia di
        % cicli nell'MBRG con le caratteristiche cercate(salveremo i
        % percorsi generati in termini di sequenza di nodi attraversati).

            % Per tutte le etichette componenti il percorso in analisi
            for q=1:leghtp
                [sizePATH,lenghtPATH]=size(PATH{1,1});
                % Azzero il contatore dei percorsi aggiuntivi ai
                % "sizePATH" già intrapresi
                count=0;
                cl=ones(1,sizePATH);
                for qq=1:sizePATH

                    % Leggo il nodo corrente
                    n_current=PATH{1,1}(qq,1);

                    % Verifico la "cardinalità" degli archi uscenti
                    [multeplicity,numOfArc]=size(MBRG{n_current,8});

                    % Azzero il contatore degli archi uscenti da
                    % n_current con la stessa etichetta
                    x=0;
                    % escludiamo le transizioni di guasto che
                    % analizziamo successivamente
                    for yy=1:numOfArc-numTclas
                        for xx=1:multeplicity
                            % Se esiste un arco etichettato con
                            % l'etichetta di interesse
                            if size(MBRG{n_current,8}{xx,yy})~=0
                                if (MBRG{n_current,8}{xx,yy}{1}==path(q))
                                    if (x==0)
                                        PATH{1,1}(qq,1)=
                                            MBRG{n_current,8}{xx,yy}{3}{1};
                                        x=1;
                                        cl(1,qq)=0;
                                    else
                                        count=count+1;
                                        PATH{1,1}(sizePATH+count,1)=
                                            MBRG{n_current,8}{xx,yy}{3}{1};
                                        PATH{1,2}(sizePATH+count,:)=
                                            PATH{1,2}(qq,:);
                                        cl(1,count+sizePATH)=0;
                                    end
                                end
                            end
                        end
                    end
                end
```

```
% cancello i percorsi terminanti in nodi in cui non é abilitata
% la transizione considerata
        for qq=sizePATH:-1:1
            if cl(1,qq)==1
                PATH{1,1}(qq,:)=[];
                PATH{1,2}(qq,:)=[];
            end
        end
        [sizePATH,lenghtPATH]=size(PATH{1,1});

%************************************************************
%   E' Necessario inserire tutti quei nodi raggiungibili con
%   transizioni silenziose a partire dai nodi raggiunti
%************************************************************
    for qq=1:sizePATH
        currentnode=PATH{1,1}(qq,1);
        [NN,hit] =faultreachability(MBRG,currentnode);
        [sizeNN,lenghtNN]=size(NN);
        countf=0;
        % se vengono raggiunti nodi ulteriori:
        if(lenghtNN>1)
            for iii=1:sizeNN
                for jjj=2:lenghtNN
                    current_node=NN(iii,jjj);
        % Se il nodo raggiunto non é ancora presente tra quelli
        % raggiungibili dal nodo iniziale, creo un nuovo
        % percorso

                    founded=find(PATH{1,1}==current_node);
                    [sizefounded,lenghtfounded]=size(founded);

                    if sizefounded==0
                        countf=countf+1;
                        PATH{1,1}(sizePATH+countf,1)=
                            current_node;
                        PATH{1,2}(sizePATH+countf,:)=
                            hit{iii,jjj-1}+PATH{1,2}(qq,:);
                    else
                % Se é presente, ma é raggiunto con una
                % differente dinamica di guasto, creo un nuovo
                % percorso
                        for iiii=1:sizefounded
                            if hit{iii,jjj-1}~=
                                PATH{1,2}(founded(iiii),:)
                                countf=countf+1;
                                PATH{1,1}(sizePATH+countf,1)=
                                    current_node;
                                PATH{1,2}(sizePATH+countf,:)=
                                    hit{iii,jjj-1}+PATH{1,2}(qq,:);
                            end
                        end
                    end
                end
            end
        end
    end

%**********************************
%   Transizioni di guasto verificate
%**********************************

% DOPPIONI??? VERIFICHIAMO CHE NON CE NE SIANO
```

```
        matPATH=cell2mat(PATH);
        [sizePATH,lenghtPATH]=size(PATH{1,1});
        for qq= sizePATH:-1:1
            xxx=matPATH(qq,1);
            ff=find(matPATH(:,1)==xxx);
            [sizef,lenghtf]=size(ff);
            if sizef~=0
                for fff=1:sizef
                    if matPATH(ff(fff),:)==matPATH(qq,:) & qq~=ff(fff)
                        matPATH(qq,:)=[];
                        PATH{1,1}(qq,:)=[];
                        PATH{1,2}(qq,:)=[];
                        break
                    end
                end
            end
        end
    end


% Abbiamo determinato tutti i nodi, che dovremo verificare essere
% punti di partenza per gli eventuali cicli

    [sizePATH,lenghtPATH]=size(PATH{1,1});

    PATH{1,1}(:,2)=PATH{1,1}(:,1);

    for q=1:leghtc

        [sizePATH,lenghtPATH]=size(PATH{1,1});
        % Azzero il contatore dei percorsi aggiuntivi ai "sizePATH"
        % già intrapresi
        count=0;
        cl=ones(1,sizePATH);
        for qq=1:sizePATH

        % Leggo il nodo corrente
            n_current=PATH{1,1}(qq,2);

        % Verifico la "cardinalità" degli archi uscenti
            [multeplicity,numOfArc]=size(MBRG{n_current,8});

        % Azzero il contatore degli archi uscenti da n_current con
        % la stessa etichetta
            x=0;
        % escludiamo le transizioni di guasto che analizzeremo
        % successivamente
            for yy=1:numOfArc-numTclas
                for xx=1:multeplicity
                % Se esiste un arco etichettato con l'etichetta di
                % interesse
                    if size(MBRG{n_current,8}{xx,yy})~=0
                        if (MBRG{n_current,8}{xx,yy}{1}==cycle(q))
                            if (x==0)
                                PATH{1,1}(qq,2)=
                                    MBRG{n_current,8}{xx,yy}{3}{1};
                                x=1;
                                cl(1,qq)=0;
                            else
                                count=count+1;
                                temporary=MBRG{n_current,8}{xx,yy}{3}{1};
                                PATH{1,1}(sizePATH+count,:)=
```

```
                                [PATH{1,1}(qq,1), temporary];
                            PATH{1,2}(sizePATH+count,:)=
                                PATH{1,2}(qq,:);
                            cl(1,count+sizePATH)=0;
                        end
                    end
                end
            end
        end
    end

% cancello i percorsi terminanti in nodi da cui non é abilitata
% la transizione in analisi
    for qq=sizePATH:-1:1
        if cl(1,qq)==1
            PATH{1,1}(qq,:)=[];
            PATH{1,2}(qq,:)=[];
        end
    end

    [sizePATH,lenghtPATH]=size(PATH{1,1});

%************************************************************
%   E' Necessario inserire tutti quei nodi raggiungibili con
%   transizioni silenziose a partire dai nodi raggiunti
%************************************************************
    for qq=1:sizePATH

        currentnode=PATH{1,1}(qq,2);
        [NN,hit] =faultreachability(MBRG,currentnode);
        [sizeNN,lenghtNN]=size(NN);
        countf=0;
        if(lenghtNN>1)
            for iii=1:sizeNN
                for jjj=2:lenghtNN
                    current_node=NN(iii,jjj);
        % Se il nodo raggiunto non é ancora presente tra quelli
        % raggiungibili dal nodo iniziale, creo un nuovo
        % percorso

                    founded=find(PATH{1,1}(:,2)==current_node);
                    [sizefounded,lenghtfounded]=size(founded);

                    if sizefounded==0
                        countf=countf+1;
                        PATH{1,1}(sizePATH+countf,2)=
                            current_node;
                        PATH{1,1}(sizePATH+countf,1)=
                            PATH{1,1}(qq,1);
                        PATH{1,2}(sizePATH+countf,:)=
                            hit{iii,jjj-1}+PATH{1,2}(qq,:);
                    else
                % Se é presente ma é raggiunto con una
                % differente dinamica di guasto, creo un nuovo
                % percorso
                        for iiii=1:sizefounded
                            if hit{iii,jjj-1}~=
                                PATH{1,2}(founded(iiii),:)
                                countf=countf+1;
                                PATH{1,1}(sizePATH+countf,2)=
                                    current_node;
                                PATH{1,1}(sizePATH+countf,1)=
```

```
                                        PATH{1,1}(qq,1);
                                    PATH{1,2}(sizePATH+countf,:)=
                                        hit{iii,jjj-1}+PATH{1,2}(qq,:);
                                end
                            end
                        end
                    end
                end
            end

          %**********************************
          %   Transizioni di guasto verificate
          %**********************************

          % DOPPIONI??? VERIFICHIAMO CHE NON CE NE SIANO
              matPATH=cell2mat(PATH);
              [sizePATH,lenghtPATH]=size(PATH{1,1});
              for qq= sizePATH:-1:1
                  xxx=matPATH(qq,2);
                  ff=find(matPATH(:,2)==xxx);
                  [sizef,lenghtf]=size(ff);
                  if sizef~=0
                      for fff=1:sizef
                          if matPATH(ff(fff),:)==matPATH(qq,:) & qq~=ff(fff)
                              matPATH(qq,:)=[];
                              PATH{1,1}(qq,:)=[];
                              PATH{1,2}(qq,:)=[];
                              break
                          end
                      end
                  end
              end

          end

%         Per un'unica classe di guasto,Abbiamo una matrice avente righe
%         nella forma:
%
%         PATH={[i,j],[{1,0}^r]}
%
%         se i==j allora siamo in presenza di un ciclo, non ci resta che
%         determinare due righe, aventi la prima uno zero in seconda
%         colonna e la seconda un uno nella stessa colonna, per poter
%         imporre diag(c)=0
%
%         Si noti che utilizzeremo faultycicle e regularcycle per tener
%         traccia della presenza o meno di percorsi con e senza guasto.

          faultycicle=0;
          regularcycle=0;


          [sizePATH,lenghtPATH]=size(PATH{1,1});
          for q=1:sizePATH

                  % Se siamo in presenza di un ciclo
              if PATH{1,1}(q,1)==PATH{1,1}(q,2)
                      % Se il ciclo ha avuto luogo dallo scatto di
                      % transizioni di guasto della classe c-esima
                  if PATH{1,2}(q,c)==1
                      faultycicle=1;
```

```
                    % Altrimenti in caso contrario
                elseif PATH{1,2}(q,c)==0
                    regularcycle=1;
                end
            end

                %Se abbiamo trovato entrambi i cicli cercati:
            if regularcycle==1 & faultycicle==1
                diag(c)=0;
                break
            end
        end


            % Se viene trovato un ciclo indeterminato, si passa alla
            % classe di guasto successiva, viceversa, si esplora un
            % ulteriore ciclo.
        if(diag(c)==0)

            % Salvataggio cicli indeterminati
            cycle_indeterminate{c,1}=path;
            cycle_indeterminate{c,2}=cycle;

            break
        end
     end
  end
end

 end


 if(diag==0)
     if numTclas==1
         fprintf('\nLa sola classe di guasto in
              analisi è non diagnosticabile')
         fprintf(' poiché\nesiste un ciclo indeterminato.\n\n')
     else
         fprintf('\nTutte le classi di guasto non sono diagnosticabili')
         fprintf(' poiché,\nper ciascuna di esse esiste un ciclo')
         fprintf(' indeterminato.\n\n')
     end

     for i=1:numTclas
         if diag(i)==0
             fprintf('Classe di guasto %d:\n',i)
             fprintf('\t path = %s\n',cycle_indeterminate{i,1})
             fprintf('\t cycle= %s\n\n',cycle_indeterminate{i,2})
         end
     end

     toc;
     t=toc;

     return
 end
end

if(diag==1)
fprintf('\nTutte le classi di guasto sono diagnosticabili, poichè\n')
fprintf('non è stato trovato alcun ciclo indeterminato\n')
else
```

```
    fprintf('\nLe seguenti classi di guasto non sono diagnosticabili, poiché')
fprintf('\nper ciascuna di esse esiste un ciclo indeterminato.\n\n')

for i=1:numTclas
        if diag(i)==0
            fprintf('Classe di guasto %d:\n',i)
            fprintf('\t path = %s\n',cycle_indeterminate{i,1})
            fprintf('\t cycle= %s\n\n',cycle_indeterminate{i,2})
        end
end
end

% Salvataggio del tempo totale impiegato per la computazione del programma
toc;
t=toc;

end
```

A.5 Lettura del BRG

Questa funzione mostra come ottenere passo dopo passo il grafo di raggiungibilità di base a partire dal nodo iniziale e seguendo tutte le transizioni abilitate e le marcature di base raggiunte.

```
function showBRG(T)

%showBRG: This function shows Basis Reachability Graph of a P/T System.
%
%         ****************************************************************
%                            ## SYNTAX ##
%
%         showBRG(T)
%          This function shows how obtain a Petri Net's Basis Reachability
%          Graph T step-by-step starting from the initial node and
%          following all enabled labels and relative reached nodes.
%
%         See also: MBRG, BRG, BRD, showMBRG, showBRD, diagnosability



% Verifica del fatto che il numero di argomenti in ingresso alla funzione
% sia esattamente uno. In caso Contrario viene plottato un messaggio di
% errore

ni=nargin;
error(nargchk(1,1,ni));

% Lettura del numero di nodi (n) e del numero di transizioni osservabili
% (t)

n=T{end,1};

[sizeL,LenghtL]=size(T{1,8});
t=0;
for f=1:n
    for ff=1:LenghtL
```

```
        [sizeL,LenghtL]=size(T{f,8});
        for fff=1:sizeL
            if(size(T{f,8}{fff,ff})==[0 0])
                continue
            else
                tt=T{f,8}{fff,ff}{1,2};
                    if(tt>t)
                        t=tt;
                    end
            end
        end
    end
end


% Numero di cifre utili all'enumerazione delle Marcature(ncifre) e delle
% transizioni (tcifre).
% [Se per esempio si avessero 100 marcature, si sarebbe interessati ad
% enumerare la marcatura iniziale come M000 anzichè 0]

ncifre= ceil(log10(n));
tcifre= ceil(log10(t));

% Determinazione numero di Posti della rete
[p,uno]=size(T{1,2});

% Visualizzazione numero di nodi del BRG

fprintf('\n Basis Reachability Graph node´s number n = %d\n\n',n)

for f=1:n %per tutte la marcature

    % Determinazione del numero di etichette (LenghtL) e del numero massimo
    % di transizioni associate ad un'etichetta (sizeL).
    [sizeL,LenghtL]=size(T{f,8});

    % Visualizzazione della Marcatura Corrente

    fprintf('\n#\t\b Marking M')

    if(f-1<10)
        for zz=1:ncifre-1
            fprintf('0')
        end
    elseif(f-1<100)
        for zz=1:ncifre-2
            fprintf('0')
        end
    elseif(f-1<1000)
        for zz=1:ncifre-3
            fprintf('0')
        end
    elseif(f-1<10000)
        for zz=1:ncifre-4
            fprintf('0')
        end
    end
    fprintf('%d=[',f-1)
    for ff=1:p
        fprintf('%d ',T{f,2}(ff,1))
    end
    fprintf('\b]´\t\b')
```

```
% Visualizzazione dello stato di diagnosi (Distinzione tra lo stato 0
% ed 1).
[x,y]=size(T{f,3});
fprintf('\n  x=[')
if y==0
    fprintf(' ]')
else
    for xx=1:y
        fprintf('%d ',T{f,3}(1,xx))
    end
    fprintf('\b]')
end

% Visualizzazione delle Transizioni abilitate dalla Marcatura
% considerata e del relativo vettore di giustificazione

noarc=1;

fprintf('\n\n \t\b Observable transitions enabled to fire:\n\t\b')

for ff=1:LenghtL
    for fff=1:sizeL
        [a,b]=size(T{f,8}{fff,ff});
        if a~=0
            noarc=0;
            l=T{f,8}{fff,ff}{1,1};
            t=T{f,8}{fff,ff}{1,2};
            n_next=T{f,8}{fff,ff}{1,3}{1};
            fprintf('  %s(t',l)

            if(t-1<10)
                for zz=1:tcifre-1
                    fprintf('0')
                end
            elseif(t-1<100)
                for zz=1:tcifre-2
                    fprintf('0')
                end
            elseif(t-1<1000)
                for zz=1:tcifre-3
                    fprintf('0')
                end
            elseif(t-1<10000)
                for zz=1:tcifre-4
                    fprintf('0')
                end
            end

            fprintf('%d) -> M',t)

            if(n_next-1<10)
                for zz=1:ncifre-1
                    fprintf('0')
                end
            elseif(n_next-1<100)
                for zz=1:ncifre-2
                    fprintf('0')
                end
            elseif(n_next-1<1000)
                for zz=1:ncifre-3
                    fprintf('0')
                end
```

```
                elseif(n_next-1<10000)
                    for zz=1:ncifre-4
                        fprintf('0')
                    end
                end

                fprintf('%d:\t\b e=[',n_next-1)
                [u,v]=size(T{f,8}{fff,ff}{1,3}{2});
                if v==0
                    fprintf(' ]\n')
                else
                    for c=1:v
                        fprintf('%d ',T{f,8}{fff,ff}{1,3}{2}(1,c))
                    end
                    fprintf('\b]\n')
                end
            end
        end

    end

    % Verifico che dal nodo ci siano di fatto archi uscenti
    if noarc==1
        fprintf('\t\b  None\n\t\b')
    end

    fprintf('\n\t\t*******\n')

end
```

A.6 Lettura dell'MBRG

```
function showMBRG(T)

%showMBRG: This function shows Modified Reachability Graph of a P/T System.
%
%          ************************************************************
%                         ## SYNTAX ##
%
%          showMBRG(T)
%           This function shows how obtain a Petri Net's Modified Basis
%           Reachability Graph T step-by-step starting from the initial
%           node and following all enabled labels and relative reached
%           nodes.
%
%          See also: BRG, MBRG, BRD, showBRG, showBRD, diagnosability



% Verifica del fatto che il numero di argomenti in ingresso alla funzione
% sia esattamente uno. In caso Contrario viene plottato un messaggio di
% errore

ni=nargin;
error(nargchk(1,1,ni));

% Lettura del numero di nodi (n) e del numero complessivo tra transizioni
```

```
% osservabili e transizioni di guasto (t).

n=T{end,1};

% Non ci serve il numero delle transizioni, quanto il numero massimo
% associato alle transizioni, non avendo un ordine preciso.

[sizeL,LenghtL]=size(T{1,8});
t=0;
for f=1:n
    for ff=1:LenghtL
        [sizeL,LenghtL]=size(T{f,8});
        for fff=1:sizeL
            if(size(T{f,8}{fff,ff})==[0 0])
                continue
            else
                tt=T{f,8}{fff,ff}{1,2};
                    if(tt>t)
                        t=tt;
                    end
            end
        end
    end
end

% Numero di cifre utili all'enumerazione delle Marcature(ncifre) e delle
% transizioni (tcifre).
% [Se per esempio si avessero 100 marcature, si sarebbe interessati ad
% enumerare la marcatura iniziale come Møøø anzichè Mø]

ncifre= ceil(log10(n));
tcifre= ceil(log10(t));

% Determinazione numero di Posti della rete
[p,uno]=size(T{1,2});

% Visualizzazione numero di nodi del BRG

fprintf('\n Modified Basis Reachability Graph node´s number n = %d\n\n',n)

for f=1:n %per tutte la marcature

    % Determinazione del numero di etichette (LenghtL) e del numero massimo di
    % transizioni associate ad un'etichetta (sizeL).
    [sizeL,LenghtL]=size(T{f,8});

    % Visualizzazione della Marcatura Corrente

    fprintf('\n#\t\b Marking M')

    if(f-1<10)
        for zz=1:ncifre-1
            fprintf('0')
        end
    elseif(f-1<100)
        for zz=1:ncifre-2
            fprintf('0')
        end
    elseif(f-1<1000)
        for zz=1:ncifre-3
            fprintf('0')
        end
```

```
elseif(f-1<10000)
    for zz=1:ncifre-4
        fprintf('0')
    end
end

fprintf('%d=[',f-1)
for ff=1:p
    fprintf('%d ',T{f,2}(ff,1))
end
fprintf('\b]´\t\b')

% Visualizzazione dello stato di diagnosi (Distinzione tra lo stato ø
% ed 1).

[x,y]=size(T{f,3});
fprintf('\n  x=[')
if y==0
    fprintf(' ]')
else
    for xx=1:y
        fprintf('%d ',T{f,3}(1,xx))
    end
    fprintf('\b]')
end



% Visualizzazione delle Transizioni abilitate dalla Marcatura
% considerata e del relativo vettore di giustificazione

fprintf('\n\n \t\b Observable and faulty transitions enabled to fire:\n\t\b')

% Variabile utile per la verifica del fatto che un nodo sia o meno "morto"
noarc=1;

for ff=1:LenghtL
    for fff=1:sizeL
        [a,b]=size(T{f,8}{fff,ff});
        if a~=0
            noarc=0;
            l=T{f,8}{fff,ff}{1,1};
            t=T{f,8}{fff,ff}{1,2};
            n_next=T{f,8}{fff,ff}{1,3}{1};

            if (l==0)
                fprintf('  eps')
            else
                fprintf('  %s(t',l)
            end

            if(t-1<9)
                for zz=1:tcifre-1
                    fprintf('0')
                end
            elseif(t-1<99)
                for zz=1:tcifre-2
                    fprintf('0')
                end
            elseif(t-1<999)
                for zz=1:tcifre-3
                    fprintf('0')
```

```
                    end
                elseif(t-1<9999)
                    for zz=1:tcifre-4
                        fprintf('0')
                    end
                end

                if (l==0)
                    fprintf('%d  -> M',t)
                else
                    fprintf('%d) -> M',t)
                end

                if(n_next-1<10)
                    for zz=1:ncifre-1
                        fprintf('0')
                    end
                elseif(n_next-1<100)
                    for zz=1:ncifre-2
                        fprintf('0')
                    end
                elseif(n_next-1<1000)
                    for zz=1:ncifre-3
                        fprintf('0')
                    end
                elseif(n_next-1<10000)
                    for zz=1:ncifre-4
                        fprintf('0')
                    end
                end

                % C'è la possibilità che il numero di transizioni regolari
                % sia nullo e che, dunque, l'insieme delle giustificazioni
                % sia l'insieme vuoto Ø.
                [u,v]=   size(T{f,8}{fff,ff}{1,3}{2});
                if(v==0)
                    fprintf('%d:\t\b e=[]\n',n_next-1)
                else
                    fprintf('%d:\t\b e=[',n_next-1)
                    for c=1:v
                        fprintf('%d ',T{f,8}{fff,ff}{1,3}{2}(1,c))
                    end
                    fprintf('\b]\n')
                end
            end
        end

    end

% Verifico che dal nodo ci siano di fatto archi uscenti
    if noarc==1
        fprintf('\t\b  None\n\t\b')
    end

    fprintf('\n\t\t*******\n')
end
```

A.7 Lettura del BRD

```
function showBRD(T)

%showBRD: This function shows Basis Reachability Diagnoser of a P/T System.
%
%         ***************************************************************
%                              ## SYNTAX ##
%
%         showBRD(T)
%          This function shows how obtain a Petri Net's Basis Reachability
%          Diagnoser T step-by-step starting from the initial node and
%          following all enabled labels and relative reached nodes.
%
%         See also: MBRG, BRG, BRD, showMBRG, showBRG, diagnosability



% Verifica del fatto che il numero di argomenti in ingresso alla funzione
% sia esattamente uno. In caso Contrario viene plottato un messaggio di
% errore

ni=nargin;
error(nargchk(1,1,ni));

% Lettura del numero di nodi (n)
n=T{end,1};

% Lettura del numero di Marcature (m).
M=T(:,5);
M=cell2mat(M);
m=max(M);

% Numero di cifre utili all'enumerazione di Marcature(ncifre) e nodi.
% [Se per esempio si avessero 100 marcature, si sarebbe interessati ad
% enumerare la marcatura iniziale come Møøø anzichè Mø]

ncifre= ceil(log10(n));
mcifre= ceil(log10(m));

% Determinazione numero di Posti della rete (p) e del numero di classi di
% guasto(numTclas)
p=-T{1,2}(1);
[tot1,tot2]=size(T{1,2});
numTclas=(tot2-p)/2;

% Visualizzazione numero di nodi del BRG

fprintf('\n Basis Reachability Diagnoser node´s number N = %d\n\n',n)

for f=1:n %per tutti i nodi

    % Visualizzazione del nodo Corrente

    fprintf('\n#\t\b Node N')

    if(f-1<10)
        for zz=1:ncifre-1
            fprintf('0')
        end
```

```
elseif(f-1<100)
    for zz=1:ncifre-2
        fprintf('0')
    end
elseif(f-1<1000)
    for zz=1:ncifre-3
        fprintf('0')
    end
elseif(f-1<10000)
    for zz=1:ncifre-4
        fprintf('0')
    end
end

fprintf('%d',f-1)

% Visualizzazione dello stato di diagnosi
fprintf('\t Delta=[')
if numTclas==0
    fprintf(' ]')
else
    for fff=1:numTclas
        fprintf('%d ',T{f,3}(1,fff))
    end
    fprintf('\b]')
end

% Determinazione del numero di Marcature (sizeN) associate al nodo.
[sizeN,LenghtN]=size(T{f,2});

fprintf('\n\n \t\b Basis Markings and corresponding X-fault and H:\t\b')
for ff=1:sizeN

    if(T{f,2}(ff,1)>-1)
        % Visualizzazione della marcatura ff-esima del nodo e numero mm
        % nel BRG/MBRG)
        mm=T{f,5}(ff,1);
        fprintf('\n\t\b  Marking M')

        if(mm-1<9)
            for zz=1:ncifre-1
                fprintf('0')
            end
        elseif(mm-1<99)
            for zz=1:ncifre-2
                fprintf('0')
            end
        elseif(mm-1<999)
            for zz=1:ncifre-3
                fprintf('0')
            end
        elseif(mm-1<9999)
            for zz=1:ncifre-4
                fprintf('0')
            end
        end
            fprintf('%d=[',mm)

        for fff=1:p
            fprintf('%d ',T{f,2}(ff,fff))
        end
        fprintf('\b]´\t\b')
```

```
        % Visualizzazione di X-Fault (Distinzione tra lo stato 0 ed 1).

        fprintf('    x=[')
        for fff=p+1:p+numTclas
            fprintf('%d ',T{f,2}(ff,fff))
        end
        fprintf('\b]')

        % Visualizzazione di H.

        fprintf('    h=[')
        for fff=p+1+numTclas:p+2*numTclas
            if(T{f,2}(ff,fff)==-1)
                fprintf('N ')
            elseif(T{f,2}(ff,fff)==-2)
                fprintf('F ')
            end
        end
        fprintf('\b]')

    end
end
% Visualizzazione delle Transizioni abilitate dal nodo considerato
% e del relativo nodo raggiunto

% Variabile utile per la verifica del fatto che un nodo sia o meno
% "morto"
noarc=1;

fprintf('\n\n \t\b Observable transitions enabled to fire:\n\t\b')
[sizeT,lenghtT]=size(T{f,4});
for ff=1:lenghtT
    [a,b]=size(T{f,4}{ff});
    if a~=0
        noarc=0;
        l=T{f,4}{1,ff}{1,1};
        n_next=T{f,4}{1,ff}{1,2};

        fprintf('  %s  -> N',l)

        if(n_next-1<10)
            for zz=1:ncifre-1
                fprintf('0')
            end
        elseif(n_next-1<100)
            for zz=1:ncifre-2
                fprintf('0')
            end
        elseif(n_next-1<1000)
            for zz=1:ncifre-3
                fprintf('0')
            end
        elseif(n_next-1<10000)
            for zz=1:ncifre-4
                fprintf('0')
            end
        end

        fprintf('%d\n',n_next-1)

    end
```

```
    end

% Verifico che dal nodo ci siano di fatto archi uscenti
    if noarc==1
        fprintf('\t\b  None\n\t\b')
    end

    fprintf('\n\t\t*******\n')
end
```

A.8 Diagnosi on-line

```
function       diagnosis(Pre, Post, M0, w, F, L, E)

%   diagnosis: This function computes diagnosis on-line for a generic Petri
%              net system.
%
%              ********************************************************
%
%                                   ## SYNTAX ##
%
%              diagnosis(Pre, Post, M0, w, F, L, E)
%
%              Given a Petri net with its matrices Pre and Post,the initial
%              marking M0, the matrix F, that indicates the Fault class,
%              the matrix L, that indicates the Labeling function, the
%              Matrix E, that indicates the Alphabet of label, and the word
%              w, this function returns a matrix where:
%              - in the first column, there is the actually fired suffix of
%                the considered word(w);
%              - in the second column, there is the corresponding Diagnosis
%                states.
%
%              F must be a cell array that has as many rows as the number
%              of fault classes, that contains in each row the fault
%              transitions that belong to the corresponding fault class.
%              L must be a cell array that has as many rows as the
%              cardinality of the considered alphabet, that contains in
%              each row the observable transitions having the same label.
%              E must be a cell array that contains in each row a string of
%              characters, each one corresponding to a different label in
%              the considered alphabet. Obviously, the cell array E is
%              ordered according to L.
%
%              The Diagnosis State for the i-th fault class is equal to:
%
%              - 0   If the ith fault cannot have occurred, because none of
%                    the firing sequence consistent with the observation
%                    contains fault transitions of class i;
%
%              - 1   If a fault transition of class i may have occurred,
%                    but is not contained in any justification of w;
%
%              - 2   If a fault transition of class ith may have occurred,
%                    but is contained in one (but not in all) justication
%                    of w;
%
```

```
%              - 3   If the i-th fault must have occurred, because all
%                    firable sequences consistent with the observation
%                    contain at least one fault transition of class i.
%
%              NB:
%              To distinguish between the case 0 and 1 we resolve the
%              following constraits set :
%                _
%               |  M +Cu*z>=0
%              <
%               |_ z(tf)>0
%
%              Where M is whichever Marking obtained by the firing of the
%              suffix i-w and tf is a fault transition.
%
%              If the set admit solution the element will be 1, otherwise
%              the element will be 0.
%
%
%              See also: BRG, showBRG, diagnosability


%   /_____________ PREPARAZIONE DATI_____________/

% Determinazione numero di posti (np), di transizioni (n), del numero di
% label e di classi di guasto:
[np,n]=size(Post);
[c,p]=size(M0');
[numLabel,lenghtL]=size(L);
[numTclas,b]= size(F);


% Determinazione del numero di transizioni osservabili (no) e non
% osservabili(nu), in particolare di quelle regolari (nreg) e di guasto
% (nf):

no=0;
for i=1:numLabel
    [sizeLi,lenghtLi]=size(L{i,1});
    no=no+lenghtLi;
end


nf=0;
for i=1:numTclas
    [sizeFi,lenghtFi]=size(F{i,1});
    nf=nf+lenghtFi;
end

nreg=n-no-nf;

nu=nreg+nf;


% Poichè le transizioni possono essere ordinate in maniera arbitraria, le
% enumeriamo in maniera da avere prima le transizioni osservabili e,
% successivamente, quelle regolari e di guasto.

% NUMBERS è un Vettore utilizzato per tenere traccia dell'originaria
% enumerazione delle transizioni.
% Es: if numbers(i)=j then la transizione tj è stata numerata ti
numbers=zeros(1,n);
```

```
% Definiamo un vettore binario tale che:
%(reg(i)==1) se la transizione i-esima è una transizione regolare.
% Infine un contatore utile per risettare le matrici definenti funzione di
% etichettatura e classi di guasto

MPre=[];
MPost=[];
MF=[];
ML=[];

reg=ones(1,n);
count=0;
for i=1:numLabel
    [sizeLi,lenghtLi]=size(L{i,1});
    for j=1:lenghtLi
        count=count+1;
        t=L{i,1}(1,j);
        reg(t)=0;
        numbers(count)=t;
        MPre=[MPre,Pre(:,t)];
        MPost=[MPost,Post(:,t)];
        ML{i,1}(1,j)=count;
    end
end

for i=1:numTclas
    [sizeFi,lenghtFi]=size(F{i,1});
    for j=1:lenghtFi
        count=count+1;
        f=F{i,1}(1,j);
        reg(f)=0;
        numbers(count)=f;
        MPre=[MPre,Pre(:,f)];
        MPost=[MPost,Post(:,f)];
        MF{i,1}(1,j)=count;
    end
end

for i=1:n
    if (reg(i)==1)
        count=count+1;
        numbers(count)=i;
        MPre=[MPre,Pre(:,i)];
        MPost=[MPost,Post(:,i)];
    end
end

Pre=MPre;
Post=MPost;
L=ML;
F=MF;

%   /_____________ START COMPUTATION_____________/

% Impostiamo la marcatura corrente pari a quella iniziale della rete e la
% giustificazione relativa pari al vettore nullo.
%
% Mold e Jold sono rispettivamente le matrici contenenti le marcature di
% base raggiunte con il suffisso in analisi e le giustificazioni associate.
% Mnew e Jnew
Mnew=[];
```

```
Jnew=[];
Mold=M0';
Jold=zeros(1,nu);

% Determinazione dello stato di diagnosi iniziale
%Sol =constrainset(Pre, Post, M0, nu, F);
% In alternativa usare:
Sol =faultclass(Pre , Post , M0, nu , 0 ,F);

% Inizializzazione della matrice dati relativ alla diagnosi
D{1,1}='eps';
D{1,2}=Sol;

% Verifica dimensioni parola osservata
[sizew,lenghtw]=size(w);

for i=1:lenghtw
% Leggo l'etichetta attuale
    label=w(i);
    transition=[];

    % inizializzo un contatore delle coppie (M,y) che andremo a determinare
    countM=1;
    % vettore utilizzato per verificare che la label possa scattare o meno
    % da una data marcatura.
    enabled=[];
    % Determino quali transizioni siano associate all'etichetta corrente
    for ii=1:numLabel
        if E{ii}==label
            transition=L{ii};
            [sizeTransition,lenghtTransition]=size(transition);
        end
    end

% Per ciascuna delle marcature di base raggiunte, verifico le marcature
% di base raggiungibili con lo scatto di tutte le transizioni t tali
% che L(t)=w(i), dunque quelle transizioni contenute in "transition".
    [sizeM,lenghtM]=size(Mold);

    % Per tutte le transizioni associate all'etichetta osservata
    for ii=1:lenghtTransition
        t=transition(ii);
        % A partire da tutte le marcature di base precedentemente raggiunte
        for iii=1:sizeM
            Mcurrent=Mold(iii,:);
            Jcurrent=Jold(iii,:);
            Mw=Mbasis(Pre,Post,Mcurrent',nu,t);


            % La funzione Mbasis è scritta in maniera INCORRETTA, dunque
            % verifico che "t" sia effettivamente abilitata dalla marcatura
            % corrente, ossia che la marcatura prodotta non abbia elementi
            % negativi.
            if all(Mw{2,1}>-1)==0
                % fprintf('la transizione %d non è abilitata\n\n',t)
                enabled(ii)=0;
                continue
            else
                enabled(ii)=1;
            end

            % La marcatura é abilitata, dunque, verifico la marcatura
```

```
            % raggiunta e la giustificazione corrispondente
            % Mnext è l'insieme delle marcature, matrice n*numero posti
            Mnext=Mw{2,1};
            [sizeJnext,lenghtJnext]=size(Mw{2,2});

            for iiii=1:sizeJnext
                Jnext=Mw{2,2}{iiii,1};
                Mnew(countM,:)=Mnext(iiii,:);
                Jnew(countM,:)=Jnext+Jcurrent;
                countM=countM+1;
            end
        end
    end

    % Se nessuna transizione è abilitata, la parola non può essere stata
    % osservata, dunque il processo di diagnosi è TERMINATO:
if enabled==0
        LastMarking=Mcurrent;
        labelUnfired=label;
        break
end

    % Ora abbiamo un insieme di coppie (marcatura - giustificazion) sul
    % quale effettueremo il calcolo dello STATO DI DIAGNOSI

    % Per ciascuna classe di guasto
    for ii=1:numTclas
        Tfi=F{ii};
        TEMP=(Jnew(:,Tfi-no));
        % estratti da tutti i j-vector gli elementi relativi alla classe
        % di guasto in analisi, non ci resta che determinare lo stato di
        % diagnosi
        if(TEMP==0)
            %Sol=constrainset(Pre,Post,M0,nu,F);
            % in alternativa usare:
            Sol =faultclass(Pre , Post , M0, nu , 0 ,F);

            % Se il problema di programamzione lineare non ha soluzione,
            % allora:
            if(Sol(1,ii)==0)
                StatoDiDiagnosi(1,ii)=0;
            % In caso contrario invece:
            else
                StatoDiDiagnosi(1,ii)=1;
            end
        % Se tutte le occorrenze  delle giustificazioni in relazione alla
        %b classe di guasto i-esima sono positive, allora:
        end
        [sizeTEMP,lenghtTEMP]=size(TEMP);
        for iii=1:lenghtTEMP
            if TEMP(:,iii)>0
                StatoDiDiagnosi(1,ii)=3;
                break
            else
                StatoDiDiagnosi(1,ii)=2;
            end
        end
    end

    % Scriviamo i dati sulla Matrice di Celle D output della funzione

    if(i==1)
```

```
        D{i+1,1}=w(i);
    else
        D{i+1,1}=[D{i,1},w(i)];
    end
    D{i+1,2}=StatoDiDiagnosi;

    Jold=Jnew;
    Mold=Mnew;
    Mnew=[];
    Jnew=[];
    StatoDiDiagnosi=zeros(1,numTclas);
end


%   /_____________ Visualizzazione dei risultati a video_____________/

[sizeD,lenghtD]=size(D);

fprintf('\n\n The following results summarize the step of the on-line')
fprintf('\n diagnosis carried out by the observed word:')
fprintf('\n\t- "%s".\n\n\n',w')

fprintf('')
fprintf('\tSUFFIX\t\t\t\t\tDIAGNOSIS STATE\n\n')
for i=1:sizeD
    fprintf('\t %s\t\t\t\t',D{i,1})
    if i==1
        fprintf('\t\t')
    elseif i<4
        fprintf('\t\t\t')
    elseif i<8
        fprintf('\t\t')
    elseif i<12
        fprintf('\t')
    elseif i>15 & i<20
        fprintf('\b')
    elseif i>19
        fprintf('\b\b')
    end
    fprintf('%d ',D{i,2})
    fprintf('\n')
end

if sizeD~=lenghtw+1
    % Se nessuna transizione è abilitata, la parola non può essere stata
    % osservata, dunque il processo di diagnosi è TERMINATO
    fprintf('\n\n- WARNING')
    fprintf('\n  The on-line diagnosis has been terminated because\n')
    fprintf('  transition "%c" cannot fire from the marking\n  [',labelUnfired)
    fprintf('%d ',LastMarking)
    fprintf('\b].\n\n')
end


end
```

A.9 Calcolo della marcature di base consistenti ad una data osservazione

```
function [ Mw ] = Mbasis(Pre , Post , M0, nu , w )


%  Mbasis: This function computes for any observation w, a set, that we
%          denote M(w), whose elements are couple of basis marking and
%          relative j-vectors
%          ************************************************************
%
%                                ## SINTAX ##
%          [ Mw ] = Mbasis(Pre, Post, M0, nu, w)
%          Given a Petri net with its matrices Pre and Post, a initial
%          marking M0, a firing sequence of observable transition w and the
%          number of  unobservable transitions nu, this function returns a
%          cell-matrix Mw that contents many row as the length of the
%          firing sequence +1, and for each row, three columns.
%          The first column is a matrix that represents the basis marking
%          consistent with the firing transition which is represented in
%          the third column. The second column of each row is an other
%          cell-matrix of one column where, the number of row is equal
%          to the number of  basis marking for this transition.
%          In each row of is sub cell-matrix there is a matrix that
%          represents the set of j-vectors corresponding to the basis
%          marking in the first column.
%          The firing sequence w must be a row array of integer where each
%          number represents the relative observable transition.
%          The matrices Pre and Post must contain in the first columns the
%          observable transitions and then that unobservable.



 [sizew,lenghtw]=size(w);
 [pn,tn]=size(Pre);
 [x,c]=size(M0);

 % verification of the input data
if (size(Post)==size(Pre))&(c==1)&(pn==x),
elseif size(Post)~=size(Pre)
        fprintf('\n ERROR!! Matrices Pre and Post have different dimension!\n')
        fprintf('\n Insert again the data\n')
else
        fprintf('\n ERRORE! dimensions of the  marking are wrong!\n')
        return
end

if(sizew~=1)
    fprintf('\n ERROR! w must be a row array\n')
    return
end

C=Post-Pre;
Cu=(C(:,(tn-nu+1):tn));

% Inizializzazione della Matrice dati. Prima riga corrispondente alla
% marcatura iniziale, giustificazione nulla corrispondente alla parola
% vuota e parola vuota stessa (indicata con pedice nullo).
% Sia M(w)={M0 0}.
```

```
Mw = {M0' {zeros(1,nu)} [0]};

% Per ciascuna delle transizioni appartenenti alla parola data
for t=1:lenghtw
    % Prelevo la transuizione da far scattare
    transition=w(t);
    Mw{t+2,2}=[];

    % Ad ogni nuova transizione,  svuoto l'insieme delle marcature
    % raggiunte con la parola w't a partire dall'insieme delle marcature
    % del passo preendente
    M=[];
    JNEW=[];
    Jnew=[];
    % 'm' è il numero di basis marking in Mw corrispondenti w',
    % (dove w=w't t=w(j) )
    Mold=Mw{t,1};
    [sizeMold,lenghtMold]= size(Mold);

    % Per ciascuna delle m marcature di base raggiunte con w'
    for i=1:sizeMold
        % verifico il numero r di vettori di giustificazione che gli
        % corrispondono.
        Mold_i=Mold(i,:);
        Jold=Mw{t,2}{i,1};
        [sizeJold,lenghtJold]= size(Jold);



        % Calcolo tutte le giustificazioni possibili per la transizione
        % j-esima.
        % 'B' è una matrice di dimensioni [c x d] dove: 'c' è il numero di
        % giustificazioni e 'd' il numero di transizioni non osservabili.

        Jcurrent = miny(Pre,Post,Mold_i',transition,nu);
        [sizeJcurrent,lenghtJcurrent]= size(Jcurrent);
        % Se B è una matrice vuota, allora la transizione j-esima non è
        % abilitata dalla marcatura in analisi.
        if ( isempty(Jcurrent))
            % Impongo che sia un vettore nullo come se la transizione fosse
            % abilitata senza l'ausilio di transizioni non osservabili
            Jcurrent=zeros(1,nu);
        end

        % Per tutte le giustificazioni minime calcolate per lo scatto di
        % t-j,
        for g=1:sizeJcurrent
            Jcurrent_g=Jcurrent(g,:);
            % calcolo le marcature raggiunte con lo scatto delle
            % giustificazioni stesse e la transizione j-esima in analisi.

            %***************************NB*******************************
            %
            % 1) Si faccia attenzione al fatto che in questa maniera
            %    vengono abilitate anche tutte quelle transizioni "cappio"
            %    nonostante il posto corrispondente non sia marcato.
            % 2) Si noti che la richiesta di scatto di transizioni non
            %    abilitate, porterebbe alla scrittura di marcature
            %    negative, invece che all'uscita anzitempo del programma.
            %
            %   Di tutti questi errori è tenuto conto nelle funzioni BRG ed
            %   MBRG.
```

```
        %
        %************************************************************

        HIT =0;
        Mcurrent = Mold_i + (Cu*Jcurrent_g')' + (C(:,transition))';
        Jnew=[];
        for k=1:sizeJold
            Jnew(k,:)=Jold(k,:)+Jcurrent_g;
        end

        % Verifico se la Marcatura Mcurrent sia presente nell'insieme M
        % delle marcature marcature consistenti con w'
        [sizeM,lenghtM]= size(M);

        % Se l'insieme delle marcature consistenti con w' è un insieme
        % vuoto, allora lo iniziualizziamo pari ad Mcurrent
        if sizeM==0
            M=Mcurrent;
            JNEW{1,1}=Jnew;

        % in caso contrario, verifichiamo che la marcatura sia già
        % stata raggiunta in passato
        else
            for k=1:sizeM
                if Mcurrent==M(k,:)
                    HIT=1;
                    JNEW{k,1}=[JNEW{k,1};Jnew];
                    break
                end
            end

            if HIT ==0
                M(end+1,:)=Mcurrent;
                JNEW{end+1,1}=Jnew;
            end
        end
        [sizeM,lenghtM]= size(M);
    end
end

% Per ogni marcatura consistente con la parola w' è necessario
% eliminare le giustificazioni ridondanti:
[sizeM,lenghtM]= size(M);
% Per tutti i gruppi di giustificazioni
for i=1:sizeM
    % Per tutte le giustificazioni presenti nel gruppo scelto
    [sizetemp,lenghttemp]= size(JNEW{i,1});
    for ii=1:sizetemp

        if ii> sizetemp
            break
        end

        temp=JNEW{i,1}(ii,:);

        for iii=sizetemp:-1:1
            if temp==JNEW{i,1}(iii,:) & ii~=iii
                JNEW{i,1}(iii,:)=[];
            end
        end

        [sizetemp,lenghttemp]= size(JNEW{i,1});
```

```
        end
    end

    % Registro l'insieme delle marcature raggiunte, le giustificazioni ed
    % il suffisso w' corrispondente
    Mw{t+1,1}=M;
    Mw{t+1,2}=JNEW;
    Mw{t+1,3}=[Mw{t,3},transition];
end

Mw=Mw(1:length(Mw)-1,:);

end
```

A.10 Modello analizzato nel capitolo 6

```
 function [Pre,Post,M0,L,E,F] = model(m,l,d,ofc)

% model:  This function builds the model of Petri net considered in the
%         thesis.
%
%         ****************************************************************
%                              ## SYNTAX ##
%
%         [Pre,Post,M0,L,E,F] = model(m,l,d,ofc)
%
%         Given the following set of parameter:
%         1) m :  the number of arc for each of two group;
%         2) l :  the number of regular transition for each arc;
%         3) d :  binary parameter such that if d==1 than the PN is
%                 diagnosable, otherwise it's not diagnosable;
%         4) ofc :binary parameter such that if ofc==1 than the PN has one
%                 only fault class, otherwise it has one fault class for
%                 each faulty transition;
%
%         this function returns
%         1) Pre & Post Incidence matrices (Pre, Post).
%         2) initial marking (M0).
%         3) labelling function (L), a cell array that has as many rows as
%            the cardinality of the considered alphabet, that contains in
%            each row the observable transitions having the same label.
%         4) alphabet of label (E), a cell array that contains in each row
%            a string of characters,each one corresponding to a different
%            label. Obviously, E is ordered according to L.
%         5) fault classes (F), a cell array that has as many rows as the
%            number of fault classes, that contains in each row the fault
%            transitions that belong to the corresponding fault class.
%
%
%         NB:
%         if ofc==1
%            the builded system has only one fault class
%         else
%            one fault class for faulty transition
%
```

```
%         See also:


% Determinazione numero di posti
np=m*(l+1)+1;

% Determinazione numero di tranzisioni
nt=1+m*l+2*(0.5*m-1)+m;

% **********************************
% Definizione della marcatura iniziale
% **********************************
M0=[m;zeros(np-1,1)];

% ****************************************
% Definizione della Matrice di Pre-incidenza
% ****************************************

nf=2*(m/2-1);

preFAULT=[
    eye(nf/2),zeros(nf/2);
    zeros(1,nf);
    zeros(nf/2),eye(nf/2);
    zeros(1,nf)
    ];

Pre=[
    m,zeros(1,nt-1);
    zeros(l*m,1),zeros(l*m,m),zeros(l*m,nf),eye(m*l);
    zeros(m,1),eye(m),preFAULT,zeros(m,l*m)
    ];


% ****************************************
% Definizione della Matrice di Post-incidenza
% ****************************************

postFAULT=[
    zeros(l*m-2-nf,nf);
    zeros(1,nf);
    eye(nf/2),zeros(nf/2);
    zeros(1,nf)
    zeros(nf/2),eye(nf/2);
    ];

Post=[
    0,ones(1,m),zeros(1,nf),zeros(1,l*m);
    ones(m,1),zeros(m,m+nf+l*m);
    zeros(m*l,1),zeros(m*l,m),postFAULT,eye(l*m)
    ];

PRE =Pre;
POST=Post;

% Ora riordiniamo le transizioni in maniera che si abbiano prima le
% transizioni osservabili, poi quelle di guasto ed infine quelle regolari.

Pre=PRE(:,1);
Post=POST(:,1);
if(d)
    for i=1:m/2
```

```
        Pre =[Pre,  PRE(:,i+1), PRE(:,m/2+i+1)];
        Post=[Post,POST(:,i+1),POST(:,m/2+i+1)];
    end
    Pre=[Pre,PRE(:,m+2:end)];
    Post=[Post,POST(:,m+2:end)];
else
    Pre =[ Pre, PRE(:,m/2+2), PRE(:,2:3), PRE(:,m/2+3)];
    Post=[Post,POST(:,m/2+2),POST(:,2:3),POST(:,m/2+3)];

    if m>4
        for i=3:m/2
            Pre =[Pre,  PRE(:,i+1), PRE(:,m/2+i+1)];
            Post=[Post,POST(:,i+1),POST(:,m/2+i+1)];
        end
    end
    Pre =[Pre,  PRE(:,m+2:end)];
    Post=[Post,POST(:,m+2:end)];
end

% ********************************************************************
% Definizione delle Matrici di celle L ed E, definenti la funzione di
% etichettatura e l'alfabeto
% ********************************************************************
L=[];
E=[];
labeled=find(Post(1,:)==1);
[sizelabeled,lenghtlabeled]=size(labeled);

E{1,1}=char(97);
L{1,1}=1;
if d
    for i=1:lenghtlabeled/2
        L{i+1,1}=labeled(1,2*i-1:2*i);
        E{i+1,1}=char(97+i);
    end
else
    L{2,1}=labeled(1,1);
E{2,1}=char(98);
    L{3,1}=labeled(1,2:4);
E{3,1}=char(99);
    if lenghtlabeled>4
        for i=3:lenghtlabeled/2
            L{i+1,1}=labeled(1,2*i-1:2*i);
            E{i+1,1}=char(97+i);
        end
    end
end

% ********************************************************************
% Definizione della Matrice di celle F definente le classi di guasto
% ********************************************************************
F=[];
if ofc==1
    for i=1:nf
        F{1,1}(1,i)=1+m+i;
    end
else
    for i=1:nf
        F{i,1}=1+m+i;
    end
end
end
```

Bibliografia

[1] A. Di Febbraro A. Giua. *Sistemi ad eventi discreti*. The McGraw-Hill Companies, 2002.

[2] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard. Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri nets. *Discrete Events Dynamical Systems*, 8:203–231, June 1998.

[3] F. Basile, P. Chiacchio, and G. De Tommasi. An efficient approach for online diagnosis of discrete event systems. *IEEE Trans. on Automatic Control*, 2008, in press.

[4] A. Benveniste, E. Fabre, S. Haar, and C. Jard. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Trans. on Automatic Control*, 48(5):714–727, May 2003.

[5] R.K. Boel and G. Jiroveanu. Distributed contextual diagnosis for very large systems. In *Proc. IFAC WODES'04: 7th Work. on Discrete Event Systems*, pages 343–348, September 2004.

[6] R.K. Boel and J.H. van Schuppen. Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers. In *Proc. WODES'02: 6th Work. on Discrete Event Systems*, pages 175–181, October 2002.

[7] M.P. Cabasino, A. Giua, and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 2008. Preliminary accepted.

[8] M.P. Cabasino, A. Giua, and C. Seatzu. Diagnosability of bounded Petri nets. In *Proc. 48th IEEE Conf. on Decision and Control*, December 2009. Submitted.

[9] M.P. Cabasino, A. Giua, and C. Seatzu. Diagnosis of discrete event systems using labeled Petri nets. In *Proc. 2nd IFAC Workshop on Dependable Control of Discrete Systems (Bari, Italy)*, June 2009. Submitted.

[10] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete-event systems. *Discrete Events Dynamical Systems*, 10(1):33–86, January 2000.

[11] M. Dotoli, M.P. Fanti, and A.M. Mangini. Fault detection of discrete event systems using Petri nets and integer linear programming. In *Proc. 17th IFAC World Congress*, 2008.

[12] S. Genc and S. Lafortune. Distributed diagnosis of discrete event systems using Petri nets. In *Proc. of the 24th ATPN*, pages 316–336, June 2003.

[13] A. Giua and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. In *Proc. 44th IEEE Conf. on Decision and Control*, pages 6323–6328, December 2005.

[14] S. Jiang and R. Kumar. Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *IEEE Trans. on Automatic Control*, 49(6):934–945, June 2004.

[15] G. Jiroveanu and R.K. Boel. Contextual analysis of Petri nets for distributed applications. In *16th Int. Symp. on Mathematical Theory of Networks and Systems (Leuven, Belgium)*, July 2004.

[16] Donald B. Johnson. Finding all the elementary circuit of a directed graph. *SIAM J. Comput.*, 4:77–84, 1975.

[17] J. Lunze and J. Schroder. Sensor and actuator fault diagnosis of systems with discrete inputs and outputs. 34(3):1096–1107, April 2004.

[18] The Mathworks. *Programming Tips*. 2009.

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

[20] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Trans. on Automatic Control*, 43(7):908–929, July 1998.

[21] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. on Automatic Control*, 40 (9):1555–1575, 1995.

[22] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. Control Systems Technology*, 4(2):105–124, 1996.

[23] Robert Tarjan. Enumeration of the elementary circuit of a directed graph. 1972.

[24] T. Ushio, L. Onishi, and K. Okuda. Fault detection based on Petri net models with faulty behaviors. In *Proc. SMC'98: IEEE Int. Conf. on Systems, Man, and Cybernetics (San Diego, CA, USA)*, pages 113–118, October 1998.

[25] S. H. Zad, R.H. Kwong, and W.M. Wonham. Fault diagnosis in discrete-event systems: framework and model reduction. *IEEE Trans. on Automatic Control*, 48(7):1199–1212, July 2003.