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

Implementazione di un nuovo algoritmo per la diagnosticabilità di reti posto/transizione mediante marcature di base

Rita Perria

Relatori: Alessandro Giua
Maria Paola Cabasino

ANNO ACCADEMICO 2009-2010

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

Implementazione di un nuovo algoritmo per la diagnosticabilità di reti posto/transizione mediante marcature di base

Rita Perria

Relatori: Alessandro Giua
Maria Paola Cabasino

ANNO ACCADEMICO 2009-2010

Dedicato ai miei genitori, ai miei fratelli e a mia zia

Ringraziamenti

Vorrei ringraziare tutte le persone che in questi anni mi hanno sempre sostenuto, incoraggiato e hanno sempre creduto in me, permettendomi di raggiungere questo traguardo.

Desidero ringraziare Professor Giua per i preziosi insegnamenti ricevuti durante questi anni di studio e per i consigli e l'aiuto ricevuto durante il lavoro di tesi.

Ringrazio inoltre Maria Paola Cabasino, che mi ha seguito costantemente durante tutta la tesi, è sempre stata disponibile a chiarire i miei dubbi e mi ha sempre incoraggiato.

Il ringraziamento più grande va alla mia famiglia senza la quale non avrei saputo e non saprei come fare. Grazie ai miei genitori Nanni e Maria e a mia Zia Mallena, che mi hanno sostenuto in ogni modo, non mi hanno mai fatto mancare nulla e hanno sempre creduto in me, incoraggiandomi nei momenti difficili. Grazie ai miei fratelli Pietro e Gianni e alla mia cognatina Graziella, che mi sono sempre stati vicino e mi hanno sempre tirato su di morale. Grazie di cuore.

Ringrazio inoltre le mie zie, i miei zii e i miei cugini, insomma tutta la mia famiglia, per avermi fatto sempre sentire la loro presenza e il loro affetto.

Ringrazio di cuore i miei tesori Eli, Vale, Stefy e Pina, siete le persone migliori che potessi incontrare. Grazie per tutto quello che avete fatto per me e che so che continuerete a fare, per tutti i consigli che mi avete dato e che continuerete a darmi. Grazie per tutti i bei momenti passati insieme.

Un grazie speciale va alla mia amichetta Lauretta. Grazie per avermi confortato e sopportato sempre soprattutto in questi mesi. Grazie per tutti i consigli, per

tutte le risate e anche per i rimproveri. Mi hai aiutato tanto. Grazie veramente di tutto. Desidero anche ringraziare Signora Cenzina, Signor Luigi e la loro famiglia, che mi hanno accolto come una figlia rappresentando per me un punto di riferimento.

Ringrazio Patrizietta, per essermi sempre stata vicino in tutti questi anni universitari, per l'aiuto e il conforto reciproco, per avermi dato sempre i consigli giusti e per tutti i momenti di allegria e spensieratezza passati insieme.

Grazie a tutti i miei colleghi, per l'aiuto reciproco in questi anni di studio e per tutti i piacevoli ricordi che porterò sempre con me.

Ringrazio le mie coinquiline Robi e Jessi, per aver sopportato i miei momenti di pazzia in questi mesi.

Per ultimo, ma non per importanza, ringrazio Fabrizio per l'affetto, la comprensione e la pazienza che mi ha dimostrato in questi mesi.

Indice

Capitolo 1

Introduzione

In questa tesi viene affrontato il problema della diagnosticabilità dei Sistemi ad Eventi Discreti (SED), ossia quei sistemi il cui comportamento dinamico è caratterizzato dall'accadimento asincrono di eventi e non dall'avanzamento del tempo.

Il concetto chiave alla base della diagnosi e della diagnosticabilità è quello di guasto. Un guasto è inteso come qualsiasi scostamento del sistema dal suo comportamento normale o comunque previsto. La diagnosi è quel processo tramite il quale vengono individuate le anomalie all'interno del sistema. A seguito di un osservazione il sistema si troverà in uno specifico stato di diagnosi, ossia: normale, di guasto o incerto. I guasti all'interno di un sistema possono nascere per diversi motivi. Ci sono tre principali fattori che portano allo studio del problema della diagnosi dei guasti:

1. i guasti sono inevitabili;
2. la diagnosi dei guasti è importante;
3. la diagnosi dei guasti è difficile.

I sistemi per la diagnosi possono essere classificati in due principali categorie: on-line e off-line ([23]).

Nella diagnosi off-line il sistema da diagnosticare non è in condizioni di normale funzionamento e si assume che esso sia in un banco di prova. La procedura di diagnosi consiste nel testare il sistema, emettendo una serie di comandi di prova e nell'osservare i risultati ottenuti identificando un insieme di stati in cui il sistema può trovarsi. La diagnosi off-line può essere considerata equivalente ad un problema di "verifica".

Nella diagnosi on-line si assume che il sistema sia in condizioni di normale funzionamento. L'obiettivo qui, come nel caso della diagnosi off-line, è quello di emettere una sequenza di comandi e di identificare in modo univoco lo stato del sistema. Tuttavia a differenza del caso di diagnosi off-line, si deve anche tener conto dell'occorrenza di possibili eventi incontrollabili durante il processo diagnostico.

L'individuazione dei guasti risulta essere un passo importante nel controllo dei sistemi complessi. Il continuo aumento della complessità dei sistemi rende necessario lo sviluppo di metodi sempre più accurati per la diagnosi e diagnosticabilità.

In questa tesi verrà esposto il problema della diagnosticabilità. Il problema della diagnosticabilità consiste nel determinare se il sistema è diagnosticabile oppure no, ossia nel caso in cui si verifichi un guasto, se il sistema sia in grado di rilevarlo in un numero finito di passi.

1.1 Stato dell'arte

Il problema della diagnosi dei guasti ha ricevuto una grande attenzione nella letteratura. Diversi e originali sono gli approcci teorici mediante automi, proposti in tal senso, si veda ad esempio [7, 13, 16, 20, 24, 32]. Recentemente il problema della diagnosi è stato affrontato anche usando una particolare classe di SED, ossia le reti di Petri. L'uso delle reti di Petri offre dei significativi vantaggi per la loro doppia rappresentazione: grafica e matematica. Tra i vari contributi apportati in questo settore possiamo citare i lavori di Ushio *et al.* [29], Aghasaryan *et al.* [3], Benveniste *et al.* [5], Jiroveanu e Boel [6, 17], Basile *et al.* [4], Genc e Lafortune [14]. La maggior parte di questi approcci richiede una completa enumerazione dello spazio di stato. In questa tesi si farà riferimento alla teoria sviluppata da Cabasino, Giua e Seatzu [9, 10, 11] relativamente alla diagnosi dei SED.

La differenza fondamentale tra gli approcci precedentamente citati, e l'approccio alla diagnosi usando reti di Petri in [9, 15] sta nel concetto di *marcatura di base*. L'approccio tramite marcatura di base permette di enumerare solo un sottoinsieme dello spazio di stato.

Per qunato riguarda la diagnosticabilità, il suo studio è stato ampiamente affrontato nell'ambito degli automi, si veda il lavoro di Sampath *et al.* in ([25, 26]).

Sono stati proposti invece, pochi risultati relativamente alle reti di Petri (vedi [8, 12, 25, 26, 29, 31]). In questa tesi faremo riferimento al lavoro di Cabasino *et al.* ([10]).

1.2 Approccio proposto

Il lavoro da cui siamo partiti per questa tesi è quello di Cabasino *et al.* [10] relativo alla diagnosticabilità delle reti di Petri limitate[1]. In questo lavoro viene presentato un approccio basato sul concetto di *marcatura di base*, che consente di rappresentare in maniera più compatta lo spazio di stato, evitando come richiesto nel caso degli automi, di dover enumerare l'intero spazio di stato.

In tale lavoro viene proposto uno studio della diagnosticabilità gravoso da un punto di vista computazionale. Questa tesi si pone l'obiettivo di proporre una metodologia alternativa per lo studio della diagnosticabilità che ne allievi la complessità. Per fare ciò abbiamo sfruttato i risultati ottenuti nell'ambito degli automi, e li abbiamo applicati al concetto di marcatura di base sviluppato per le reti di Petri.

Il punto di partenza dell'intero lavoro è stato quello di testare la metodologia proposta. Nella metodologia precedente, trattata da Cabasino *et al.* in [10], la diagnosticabilità veniva valutata tramite l'analisi di due grafi il Modified Basis Reachability Graph (MBRG) e il Basis Reachability Diagnoser (BRD). In questa tesi la diagnosticabilità viene valutata attraverso il Modified Basis Reachability Graph e il Modified Basis Reachability Diagnoser (MBRD). La differenza del MBRD rispetto al BRD è che il MBRD viene costruito considerando non solo le marcatura di base ma tutte le possibili marcature presenti nel MBRG. In entrambe le metodologie la diagnosticabilità è basata sui concetti di cicli incerti e indeterminati. Sostanzialmente l'analisi consiste nello stabilire se esistano certi

[1] Una rete di Petri è limitata quando il suo spazio di stato è finito.

cicli (che chiameremo cicli incerti) nel BRD / MBRD e se così è,verificare se sono soddisfatte determinate condizioni nel MBRG, ossia stabilire se tali cicli sono indeterminati oppure no.

Entrambe queste metodologie sono corrette, la differenza sostanziale è che nella precedente metodologia la verifica della diagnosticabilità si basava sull'analisi di tali cicli e di *tutti* i percorsi che dalla marcatura iniziale conducevano ad essi. Al contrario il nuovo metodo non va ad analizzare i percorsi che conducono ai cicli, ma solo i cicli stessi, riducendo così il tempo di calcolo. Al contrario, il contro della nuova metodologia è che il numero di marcature da considerare nella costruzione del MBRD è maggiore.

Dal momento che questa nuova metodologia ha preso spunto da quella usata per la diagnosi degli automi, essa è stata inizialmente testata attraverso la libreria UMDES [19], sviluppata per modellare gli automi a stati finiti e per la verifica di diverse proprietà tra cui la diagnosticabilità. Tale libreria è stata sviluppata dal gruppo DES (Discrete Event Systems) dell'università del Michigan sotto la coordinazione di Stèphane Lafortune e Demosthenis Teneketzis. UMDES comprende comandi per:

- manipolazione di macchine a stati;
- diagnosi dei guasti;
- controllo supervisivo.

L'automa in ingresso ad UMDES corrisponde al nostro MBRG e il diagnosticatore costruito da UMDES corrisponde al MBRD. Quindi, il primo passo è stato quello di prendere alcuni esempi di reti di Petri, costruirne il Modified Basis Reachability Graph e il Modified Basis Reachability Diagnoser. Il file di ingresso alle funzioni è il MBRG che non è altro che un automa, generato tramite la funzione *create_fsm*. La diagnosticabilità del sistema è stata valutata usando le due funzioni *diag-UR* e *dcycle*. La prima funzione calcola il diagnosticatore multi-guasto (multi-guasto indica che possono essere presenti più classi di guasto) per il sistema dato, ed è stato verificato che fosse uguale al MBRD. La seconda funzione verifica la diagnosticabilità del sistema.

Un altro problema di fondamentale importanza affrontato, è stato quello di decidere quale algoritmo utilizzare per la ricerca dei cicli in un grafo. E' stata effettuata una ricerca bibliografica che ha portato all'individuazione di vari algoritmi tra cui:

- l'algoritmo di J. C. Tiernan ([28]);
- l'algoritmo di R. Tarjan ([27]);
- l'algoritmo di H. Weinblatt ([30]);
- l'algoritmo di A. Ehrenfeucht, L. Fosdick e L. Osterwell ([1]);
- l'algoritmo di D. B. Johnson ([18]).

L'algoritmo che si è scelto di utilizzare è quello di D. B. Johnson, che si basa sui precedenti lavori di J. C. Tiernan e R. Tarjan, ma risulta essere più efficiente dal punto di vista computazionale.

A questo punto è stato implementato un toolbox per Matlab. In tale toolbox vengono forniti i programmi necessari per la costruzione del MBRD e per la verifica della diagnosticabilità. Inoltre viene fornito il programma che implementa l'algoritmo di Johnson per il calcolo dei cicli in un grafo.

Nella parte finale della tesi è stata svolta un'analisi della complessità computazionale per il calcolo delle componenti fortemente connesse di un grafo e per il calcolo dei cicli, inoltre sono state messe a confronto la metodologia precedentemente introdotta [10] e la metodologia qui proposta.

1.3 Struttura della tesi

Il lavoro di tesi è organizzato nel seguente modo.

Il capitolo 2 è dedicato alla descrizione del modello ad eventi discreti utilizzato nella tesi, ossia le reti di Petri. Il modello di reti di Petri richiamato in questo capitolo è quello delle reti posto/transizione, si tratta di un modello che non permette di rappresentare la temporizzazione degli eventi ma solo l'ordine con cui questi si verificano. Vengono richiamati alcuni concetti di base tra cui: matrici Pre e Post, marcatura, abilitazione e scatto, linguaggio di una rete e vengono poi definite alcune importanti proprietà relative alle reti di Petri.

Il capitolo 3 è incentrato sul problema della diagnosticabilità dell reti di Petri limitate. Nella prima parte del capitolo vengono richiamate alcune definizioni di base tra cui quelle di: spiegazioni minime, e-vettori, marcatura di base, diagnosticatore etc. Viene inoltre richiamata la teoria sul Modified Basis Reachability

Graph introdotta da Cabasino *et al.* in [10], e viene introdotta la definizione di Modified Basis Reachability Diagnoser, fornendo l'algoritmo per il suo calcolo. Una volta introdotte queste definizioni, vengono fornite le condizioni necessarie e sufficienti per la diagnosticabilità e viene presentato un metodo per testarla basato sull'analisi del Modified Basis Reachability Graph e del Modified Basis Reachability Diagnoser.

Nel capitolo 4 viene illustrato l'algoritmo di Johnson (si veda [18]) per la determinazione dei cicli in un grafo orientato e vengono mostrati degli esempi per chiarirne il funzionamento. La parte finale del capitolo è dedicata alla verifica della complessità computazionale dell'algoritmo di Johnson. Vengono prese in considerazioni due reti di Petri parametriche, in cui il parametro che viene fatto variare è il numero di gettoni nella marcatura iniziale. Per ognuna di queste reti viene costruito il grafo di raggiungibilità e di quest'ultimo vengono valutati il numero di nodi, il numero di archi e il numero di componenti fortemente connesse che ne fanno parte. Per ognuna delle componenti fortemente connesse trovate viene calcolato il numero di nodi, il numero di archi e infine il numero di cicli elementari. Vengono quindi analizzati i tempi di calcolo delle componenti fortemente connesse e dei cicli.

Il capitolo 5 è dedicato al Toolbox di Matlab. In esso vengono descritti i programmi che fanno parte del toolbox. Il funzionamento di tali programmi viene chiarito tramite dei semplici esempi. Le funzioni principali contenute nel pacchetto sono le seguenti:

- MBRD.m : funzione che costruisce il Modified Basis Reachability Diagnoser;
- showMBRD.m : funzione creata per rendere di più semplice lettura l'uscita della funzione MBRD.m;
- algoritmo_Johnson.m : funzione che determina i cicli elementari presenti in grafo;
- MBRD_diag_ott.m : funzione per la verifica della diagnosticabilità.

Nel capitolo 6 sono presentati alcuni esempi che hanno come scopo quello di testare il software sviluppato in questa tesi. Viene inoltre preso in considerazione un modello fisico parametrico, tale modello viene utilizzato per fare un confronto tra il metodo per la verifica della diagnosticabilità precedentemente sviluppato e il metodo trattato in questa tesi. Sono state fatte delle simulazioni

per diversi valori dei parametri e raccolti i risultati ottenuti facendo le dovute considerazioni.

Il capitolo 7 è il capitolo conclusivo della tesi.

1.4 Contributi della tesi

I contributi apportati da questa tesi sono sostanzialmente i seguenti:

- testare la correttezza della nuova metodologia proposta;
- analizzare la letteratura relativa agli algoritmi per il calcolo dei cicli in un grafo
- analizzare la complessità computazionale dell'Algoritmo di Johnson;
- implementazione della nuova procedura e creazione di un Toolbox per Matlab;
- confronto tra la metodologia precedente ([10]) e la metodologia proposta in questa tesi.

Capitolo 2

Introduzione alle reti posto/transizione

Sommario

Questo capitolo ha come obiettivo, quello di richiamare i concetti principali relativi alle reti di Petri (RdP), in particolare alle reti posto/transizione (P/T), su cui si basa l'intera tesi. Per maggiori dettagli facciamo riferimento al lavoro di A. Di Febbraro e A. Giua ([2]).

2.1 Introduzione alle reti di Petri

Le *RdP* sono un modello di *Sistemi ad Eventi Discreti* (SED) che trae origine dal lavoro di dottorato di Carl Adam Petri ([21]), ricercatore tedesco di Lipsia. Un SED è un sistema il cui comportamento dinamico è caratterizzato dall'accadimento asincrono di eventi e non dall'avanzamento del tempo come accade nei *Sistemi ad Avanzamento Temporale* (SAT). In tali sistemi infatti, il comportamento del sistema è descritto da segnali ossia funzioni della variabile indipendente tempo. L'evoluzione di tali sistemi nasce dal trascorrere del tempo.

Il modello di RdP che viene richiamato in questo capitolo, è quello delle *reti*

posto/transizione (P/T). Si tratta di un modello che non permette di rappresentare la temporizzazione degli eventi ma solo l'ordine in cui questi si verificano.

Le reti P/T sono importanti per diversi motivi:

- Sono un formalismo grafico e matematico: conseguentemente, combinano alla facilità di comprensione, la possibilità di applicare un vasto insieme di tecniche di analisi per lo studio delle proprietà di interesse.
- Consentono di rappresentare in maniera compatta anche sistemi con uno spazio di stato di dimensione elevata. Esse infatti non richiedono di rappresentare tutti i possibili valori dello stato di un sistema ma solo le regole che ne governano l'evoluzione.
- Permettono la rappresentazione esplicita del concetto di concorrenza.
- Permettono la rappresentazione modulare, ossia consentono di rappresentare ogni sottosistema con una sottorete, e successivamente combinare le diverse sottoreti per ottenere il modello del sistema complessivo

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:*

$$C = Post - Pre$$

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

Nel compattare le due matrici Pre e $Post$ per costruire la matrice di incidenza, spesso si perde qualche informazione sulla struttura della rete. Data C non è sempre possibile ricostruire il grafo, mentre date le matrici Pre e $Post$ è possibile ricostruire perfettamente il grafo. Vediamo un esempio che renderà più chiare le considerazioni precedenti.

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. ■

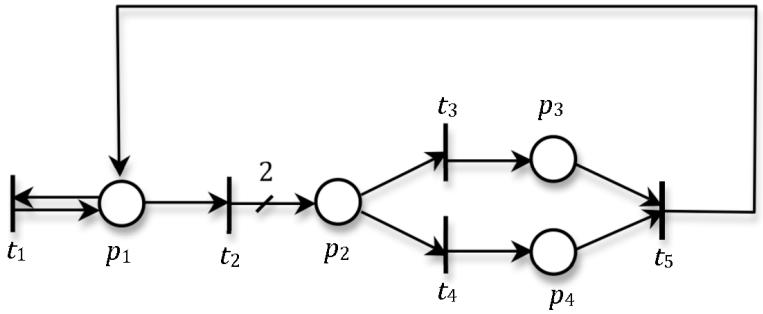


Figura 2.1: Una rete P/T

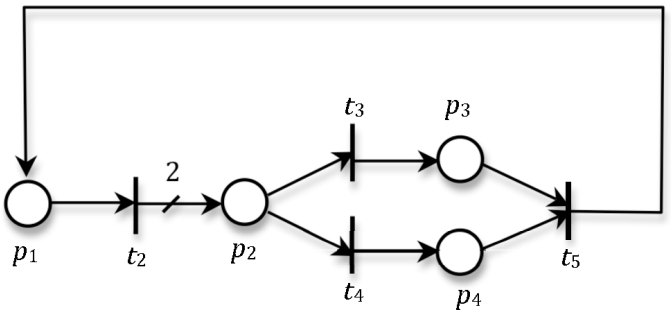


Figura 2.2: Una rete P/T ricostruita a partire dalla sua matrice di incidenza

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.
${}^\bullet p = \{t \in T \mid Post(p,t) > 0\}$: 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*

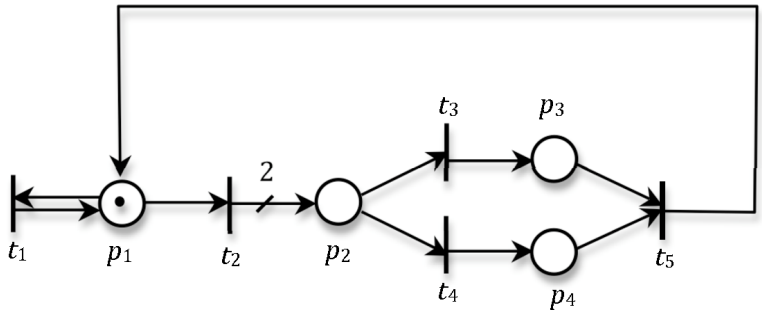


Figura 2.3: Evoluzione di una rete marcata. Marcatura iniziale.

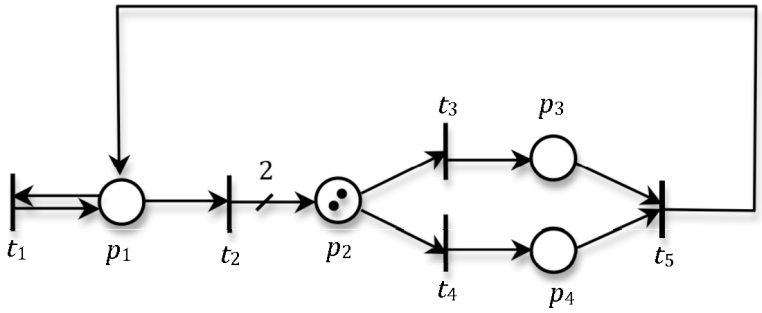


Figura 2.4: Evoluzione di una rete marcata. Marcatura raggiunta dopo lo scatto della transizione t_2.

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.

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'. Cioé 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. *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 $\sigma = t_3 t_4 t_5 t_1$ il cui scatto porta alla marcatura iniziale $M_0 = [\ 1 \quad 0 \quad 0 \quad 0\ ]^T$.

Definizione 2.4.4. *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'insieme T.

Definizione 2.4.5. *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:*

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

$\vec{\sigma}$ 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 importanti proprietà relative alle 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 un sistema di rete* $\langle N, M_0 \rangle$ *dove* N *è* 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 $\mathscr{L}(t)$ e che l'occorrenza di t dia luogo ad un'uscita osservabile $\mathscr{L}(t)$. Se la transizione fosse etichettata con la parola vuota, $\mathscr{L}(t) = \varepsilon$, 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 $\mathscr{L} : T \to E \cup \{\varepsilon\}$ *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* $G = \langle N, M_0, \mathscr{L} \rangle$ *dove* $N = (P, T, Pre, Post)$*,* M_0 *è la marcatura iniziale e* $\mathscr{L} : T \to E \cup \{\varepsilon\}$ *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* $\langle N, M_0, \mathscr{L} \rangle$ *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* $M \in R(N, M_0)$*:* $M[t\rangle \wedge M[t'\rangle \Rightarrow [\mathscr{L}(t) \neq \mathscr{L}(t')]$*. Ciò assicura che la conoscenza dello scatto di etichette* $\mathscr{L}(t)$ *sia sufficiente per ricostruire la marcatura a cui lo scatto di* t *porta.*
- λ- free*: se nessuna transizione è etichettata con la parola vuota*[1].
- Arbitraria*: se la funzione di etichettatura* $\mathscr{L}$ *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 $T_u = \{t \in T \mid \mathscr{L}(t) = \varepsilon\}$. Le transizioni di T_u sono dette *non osservabili* o *silenziose*. Indichiamo con T_o l'insieme delle transizioni etichettate con un simbolo in $\mathscr{L}$. Le transizioni di T_o sono dette *osservabili*, poiché quando scattano possono essere osservate.

[1] Nella letteratura delle RdP, la parola vuota è indicata con λ, mentre nella letteratura dei linguaggi formali 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.

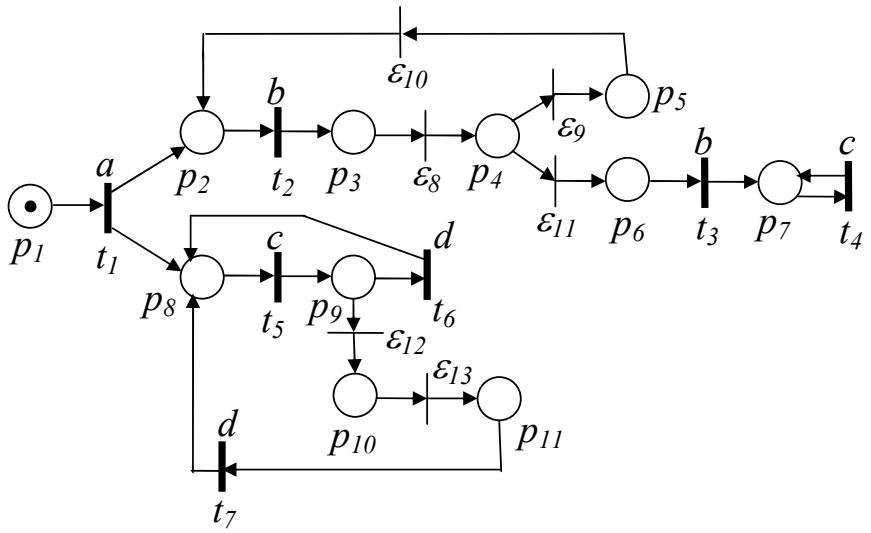


Figura 2.5: Un esempio di RdP etichettata

Si tenga presente che in questa tesi assumiamo che la stessa etichetta $l \in \mathscr{L}$ 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 $\mathscr{L}(t_1) = \mathscr{L}(t_2) = l \in \mathscr{L}$. 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 $w = \mathscr{L}(\sigma)$. 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. *Sia $\langle N, M_0 \rangle$ un sistema di RdP etichettata con funzione di etichettatura $\mathscr{L} : T \to E \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Sia $w \in L^*$ la parola osservata. Definiamo*

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

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

In altre parole, data un'osservazione w, $\mathscr{S}(w)$ è l'insieme delle sequenze che possono essere scattate. ■

Definizione 2.7.4. *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 $\mathscr{L}^{-1}(l)$ 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 $\mathscr{L}^{-1}(w)$ l'insieme delle sequenze di transizioni $\sigma \in T^*$tali che$\mathscr{L}(\sigma) = w$.

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

Diagnosticabilità delle reti di Petri limitate

Sommario

In questo capitolo ci occuperemo del problema della diagnosticabilità delle RdP limitate. In particolare verranno fornite le condizioni necessarie e sufficienti per la diagnosticabilità. Verrà presentato un metodo per testare la diagnosticabilità basato sull'analisi di due grafi dipendenti dalla struttura della rete. Questi due grafi sono rispettivamente: il *Modified Basis Reachability Graph* (MBRG) e il *Modified Basis Reachability Diagnoser* (MBRD). L'MBRG è stato trattato nel lavoro [10].

3.1 Diagnosticabilità mediante reti di Petri

In questa tesi ci concentriamo sulle RdP limitate: forniremo le condizioni necessarie e sufficienti per la diagnosticabilità e daremo un metodo sistematico per l'analisi della diagnosticabilità. Questo metodo richiede la costruzione di due grafi etichettati e orientati denominati rispettivamente: *Modified Basis Reachability Graph* (MBRG) e *Modified Basis Reachability Diagnoser* (MBRD). Il MBRG è una variante del BRG così come il MBRD è una variante del BRD.

In breve, l'analisi consiste nel determinare se nel MBRD sono presenti dei cicli detti "incerti" e nel verificare se sono soddisfatte determinate condizioni su di essi.

3.1.1 Inquadramento del problema

Assumiamo che l'insieme di transizioni sia partizionato come $T = T_o \cup T_u$ dove T_o è l'insieme di transizioni osservabili, e T_u è l'insieme di transizioni non osservabili. Quando una transizione osservabile scatta osserviamo la sua etichetta, quindi le nostre osservazioni consistono in sequenze di simboli nell'alfabeto L. L'insieme di transizioni non osservabili è suddiviso a sua volta in due sottoinsiemi, detti $T_u = T_f \cup T_{reg}$, dove T_f include tutte le transizioni di guasto mentre T_{reg} include tutte le transizioni relative a eventi non osservabili ma regolari. L'insieme T_f è ulteriormente suddiviso in r sottoinsiemi T_f^i, dove $i = 1, \ldots, r$, che modellano le differenti classi di guasto.

Definizione 3.1.1. *Un sistema di rete $\langle N, M_0 \rangle$ è detto diagnosticabile rispetto alla classe di guasto T_f^i se non esistono due sequenze σ_1 e σ_2 in T^* che soddisfano le seguenti condizioni:*

- $\mathscr{L}(\sigma_1) = \mathscr{L}(\sigma_2)$,
- $\forall\, t_f \in T_f^i,\ t_f \notin \sigma_1$,
- $\exists$ *almeno una* $t_f \in T_f^i$ *tale che* $t_f \in \sigma_2$,
- σ_2 *può essere "arbitrariamente lunga" dopo un guasto* $t_f \in T_f^i$*, cioè dato un qualunque* $p \in \mathbb{N}$ *si può sempre scegliere una sequenza* σ_2 *tale che* $|\sigma_2| > p$ *dopo il guasto.*

■

Definizione 3.1.2. *Un sistema di rete $\langle N, M_0 \rangle$ è detto diagnosticabile se è diagnosticabile in riferimento a tutte le classi di guasto.* ■

Si noti che la diagnosticabilità di un sistema non implica che siamo in grado di distinguere tra transizioni appartenenti alla medesima classe di guasto, ma semplicemente, implica che se una o più transizioni in una data classe di guasto sono scattate, allora dopo un numero finito di osservazioni siamo in grado di stabilire

che almeno una transizione di quella classe è scattata. Ci occuperemo di fornire le condizioni necessarie e sufficienti per la diagnosticabilità, in particolare considereremo sistemi di RdP etichettate che soddisfano le seguenti condizioni:

1. Il sistema di rete $\langle N, M_0 \rangle$ è *limitato*, e *non si blocca dopo lo scatto di una transizione di guasto*;
2. La sottorete $T_u - indotta$ è aciclica;
3. La funzione di etichettatura $\mathscr{L} : T \to E \cup \{\varepsilon\}$ può associare la stessa etichetta a differenti transizioni;
4. La struttura di N è nota così come la sua marcatura iniziale M_0.

3.1.2 Definizioni di base

In questo paragrafo richiamiamo alcuni definizioni di base introdotte per la prima volta nei lavori ([9]) e ([11]).

Definizione 3.1.3. *([9]) Data una marcatura M ed una transizione $t \in T_o$ definiamo*

$$\Sigma_{\min}(M,t) = \{\sigma \in T_u^* \mid M[\sigma\rangle M',\ M' \geq Pre(\cdot,t),\ \nexists\, \sigma' \mid M[\sigma'\rangle M'',\ M'' \geq Pre(\cdot,t) :\ \pi(\sigma') \lneqq \pi(\sigma)\}$$

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

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

il corrispondente insieme degli e-vettori minimi. ■

Nel caso delle RdP etichettate quello che osserviamo è l'etichetta l. Allora è utile definire i seguenti insiemi.

Definizione 3.1.4. *([11]) Data una marcatura M ed una osservazione $l \in L$ definiamo l'* insieme delle spiegazioni minime di l a M *come*

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

l'insieme delle coppie (transizione etichettata l - corrispondente e-vettore minimo) e definiamo l'insieme degli e-vettori minimi di l a M come

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

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

Data una parola $w \in L^*$ chiamiamo *giustificazione di w* la corrispondente sequenza di transizioni non osservabili intervallate con σ_0 il cui scatto abilita σ_0 e il cui vettore di scatto è minimo.

Definizione 3.1.5. *([11]) Dato il sistema di rete $\langle N, M_0\rangle$ con funzione di etichettatura $\mathscr{L} = T \to L \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Sia $w \in L^*$ una data osservazione. Definiamo*

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

l'insieme di coppie (sequenza $\sigma_o \in T_o^*$ con $\mathscr{L}(\sigma_o) = w$ - corrispondente giustificazione *di w). Definiamo*

$$\hat{Y}_{\min}(M_0, w) = \{(\sigma_o, y), \sigma_o \in T_o^*,\ \mathscr{L}(\sigma_o) = w,\ y \in \mathbb{N}_u^n \mid \exists (\sigma_o, \sigma_u) \in \hat{\mathscr{J}}(w) : \pi(\sigma_u) = \vec{y}\}$$

l'insieme di coppie (sequenza $\sigma_o \in T_o^*$ con $\mathscr{L}(\sigma_o) = w$ – corrispondente j-vettore*).* ■

In altre parole, $\hat{\mathscr{J}}(w)$ è l'insieme di coppie sequenza $\sigma_o \in T_o^*$ etichettata w - giustificazione e i vettori di scatto di queste sequenze sono detti *j-vettori*.

Definizione 3.1.6. *([11]) Dato il sistema di rete $\langle N, M_0\rangle$ con funzione di etichettatura $\mathscr{L} : T \to L \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Sia w una data osservazione e $(\sigma_o, \sigma_u) \in \hat{\mathscr{J}}(w)$ una generica coppia (sequenza di osservazioni etichettate w - corrispondente giustificazione minima). La marcatura*

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

raggiunta scattando σ_o intervallata con la giustificazione minima σ_u, è detta marcatura di base *e $\vec{y}$ è chiamato* j-vettore *(o vettore giustificazione).* ■

In altre parole, la marcatura di base M_b è la marcatura raggiunta dalla marcatura M_0 con lo scatto di w e di tutte quelle transizioni non osservabili che sono strettamente necessarie ad abilitare w.

Ovviamente, poiché in generale esiste più di una giustificazione per una parola w, la marcatura di base non è in genere unica.

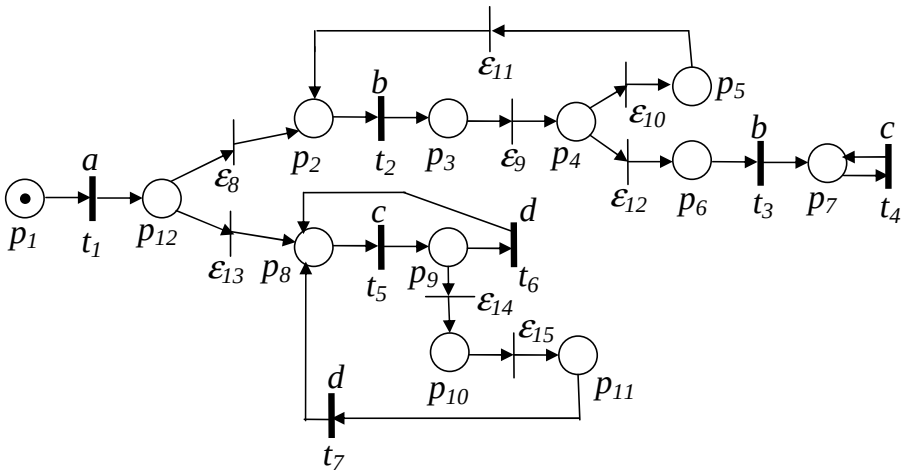


Figura 3.1: Esempio di rete di Petri.

Definizione 3.1.7. *([11]) Dato il sistema di rete $\langle N, M_0\rangle$ con funzione di etichettatura $\mathscr{L} = T \to L \cup \{\varepsilon\}$, dove $N = (P, T, Pre, Post)$ e $T = T_o \cup T_u$. Sia $w \in L^*$ una parola osservata. Definiamo*

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

l'insieme di coppie (marcatura di base - relativo j-vettore) che sono consistenti con $w \in L^*$. ■

Esempio 3.1.1. *Consideriamo la rete in Figura 3.1 dove l'insieme di transizioni osservabili è $T_o = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ e l'insieme di transizioni non osservabili è $T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{14}, \varepsilon_{15}\}$. La funzione di etichettatura è $\mathscr{L}(t_1) = a$, $\mathscr{L}(t_2) = \mathscr{L}(t_3) = b$, $\mathscr{L}(t_4) = \mathscr{L}(t_5) = c$ e $\mathscr{L}(t_6) = \mathscr{L}(t_7) = d$.*

Assumiamo $w = abb$. L'insieme di giustificazioni è $\hat{\mathscr{J}}(w) = \{(t_1 t_2 t_3, \varepsilon_8\varepsilon_9\varepsilon_{12}), (t_1 t_2 t_2, \varepsilon_8\varepsilon_9\varepsilon_{10}\varepsilon_{11})\}$ e l'insieme di j-vettori è $\hat{Y}_{min}(M_0, w) = \{(t_1 t_2 t_3, [1\ 1\ 0\ 0\ 1\ 0\ 0\ 0]^T), (t_1 t_2 t_2, [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0]^T)\}$. I j-vettori precedenti conducono alle marcature $M_1 = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^T$ e $M_2 = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$. Allora $\mathscr{M}(w) = \{(M_1, [1\ 1\ 0\ 0\ 1\ 0\ 0\ 0]^T)), (M_2, [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0]^T))\}$. ■

3.1.3 Diagnosi usando le reti di Petri

Introduciamo ora la definizione di diagnosticatore:

Definizione 3.1.8. *([11]) Un* diagnosticatore *è una funzione*

$$\Delta : L^* \times \{T_f^1, T_f^2, \ldots, T_f^r\} \to \{0,1,2,3\}$$

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* $\sigma \in \mathscr{S}(w)$ *(vedi Definizione 2.7.3) e per tutte le* $t_f \in T_f^i$ *si verifica 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 $\sigma \in \mathscr{S}(w)$ *e una* $t_f \in T_f^i$ *tali che* $t_f \in \sigma$ *ma*

  (ii) per tutte le $(\sigma_o, \sigma_u) \in \hat{\mathscr{J}}(w)$ *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* $(\sigma_o, \sigma_u), (\sigma'_o, \sigma'_u) \in \hat{\mathscr{J}}(w)$ *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* $\sigma \in \mathscr{S}(w)$ *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, poiché tutte le sequenze scattabili consistenti con l'osservazione contengono almeno un guasto in* T_f^i.

■

La precedente definizione può essere espressa come proposizione in termini di marcatura di base e di giustificazione.

Proposizione 3.1.1. *([9]) Consideriamo la parola osservata* $w \in L^*$.

- $\Delta(w, T_f^i) \in \{0,1\}$ *se e solo se per tutte le coppie* $(M, y) \in \mathscr{M}(w)$ *e per tutte* $t_f \in T_f^i$ *si verifica che* $y(t_f) = 0$.
- $\Delta(w, T_f^i) = 2$ *se e solo se esiste una* $(M, y) \in \mathscr{M}(w)$ *e* $(M', y') \in \mathscr{M}(w)$ *tale che:*
  (i) esiste $t_f \in T_f^i$ *tale che* $y(t_f) > 0$,
  (ii) per tutte $t_f \in T_f^i$, $y'(t_f) = 0$.
- $\Delta(w, T_f^i) = 3$ *se e solo se per tutte le coppie* $(M, y) \in \mathscr{M}(w)$ *esiste* $t_f \in T_f^i$ *tale che* $y(t_f) > 0$.

La seguente proposizione introduce un metodo utilizzato per distinguere tra gli stati di diagnosi 0 e 1, basato sulla risoluzione di un sistema di equazioni lineari.

Proposizione 3.1.2. *([9]) Per una RdP la cui sottorete non osservabile sia aciclica, sia* $w \in L^*$ *una parola osservata tale che per tutte le* $(M, y) \in \mathscr{M}(w)$ *si abbia* $y(t_f) = 0 \ \forall \ t_f \in T_f^i$.

Consideriamo l'insieme di vincoli:

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

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

In base ai due precedenti risultati, se la sottorete T_u-indotta è aciclica, la diagnosi può essere effettuata semplicemente osservando l'insieme $\mathscr{M}(w)$ 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 3.1.2. *Consideriamo la RdP in Figura 3.1, dove* $T_f^1 = \{\varepsilon_{12}\}$ *e* $T_f^2 = \{\varepsilon_{14}\}$.

Sia $w = abb$. $\mathscr{M}(w) = \{(M_1, [1\ 1\ 0\ 0\ 1\ 0\ 0\ 0]^T), (M_2, [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0]^T)\}$, *dove* $M_1 = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^T$ *e* $M_2 = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ *sono le*

marcature determinate nell'esempio 3.1.1. In questo caso si ha ad esempio che, $\Delta(w, T_f^1) = 2$. *Infatti,* $y_1 = [1\ 1\ 0\ 0\ 1\ 0\ 0\ 0]^T$ *contiene* $\varepsilon_1 2$ *appartenente a* T_f^1, *mentre* $y_2 = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0]^T$ *non contiene una transizione di guasto contiene* $t_f \in T_f^1$. ■

In [9] è stato mostrato che nel caso di RdP limitate un utile strumento per la diagnosi è il *Basis Reachability Graph* (BRG).

In [10] è stato mostrato come il BRG necessita di essere modificato se lo si vuole utilizzare come strumento ausiliario per stabilire se il sistema è diagnosticabile. A questo scopo è stato definito un nuovo grafo, chiamato *Modified Basis Reachability Graph* (MBRG).

3.1.4 Modified Basis Reachability Graph - 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 $\mathscr{T}(M)$ 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 all'insieme di transizioni T_{reg} e C_{reg} la restrizione della matrice di incidenza all'insieme di transizioni 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 del MBRG è il seguente.

Algoritmo 3.1.1. ***[Calcolo del MBRG]***

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

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

Non assegnare alcun tag.

2. *Sinchè 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* $t : L(t) = l \wedge Y_{\min}^{mod}(M,t) \neq \emptyset$*:*
    - *per tutte le* $e \in Y_{\min}^{mod}(M,t)$*:*
      - *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* $\forall i = 1,\dots,r$

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

      - *aggiungi un arco* $(l(t), e)$ *dal nodo* (M,x) *al nodo* (M', x')
- ***2.4.*** *per* $i = 1,\dots,r \ : \ x(T_f^i) = 1$
  - ***2.4.1.*** *per tutte le* $t_f \in T_f^i : Y_{\min}^{mod}(M,t_f) \neq \emptyset$*:*
    - *per tutte le* $e \in Y_{\min}^{mod}(M,t_f)$*:*
      - *sia* $M' = M + C_{reg} \cdot e + C(\cdot, t_f)$,
      - *se* $\nexists$ *un nodo senza* M'*:*
        - *aggiungi un nuovo nodo al grafo contenente la coppia* (M', x')*, dove* $\forall i = 1,\dots,r$

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

      - *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 $l \in L$

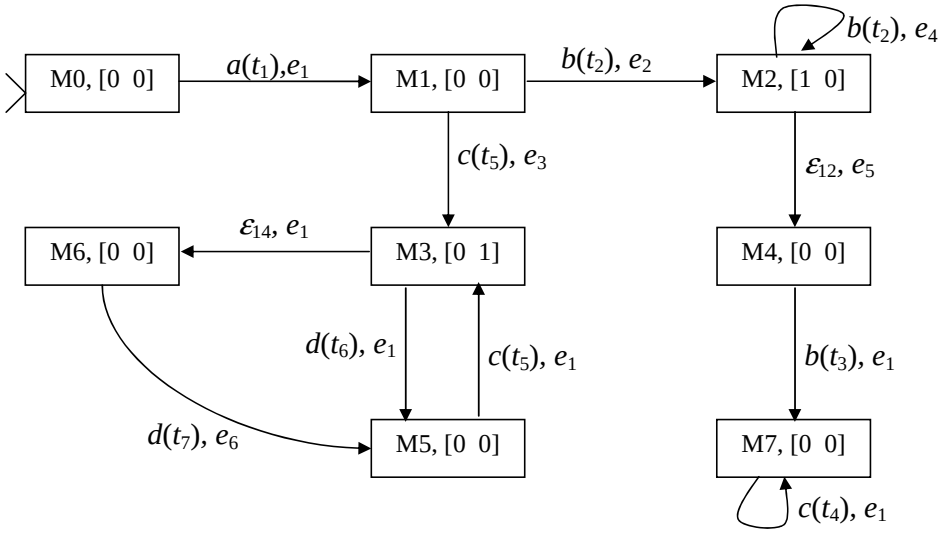


Figura 3.2: L'MBRG della RdP in Figura 3.1.

(passo 2.3) e tutte le classi di guasto $i = 1, \ldots, r$ (passo 2.4), tali che esista una transizione t con $\mathscr{L}(t) = l$ 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 $t_f \in T_f^i$, 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 $(\mathscr{L}(t), e)$ 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 3.1.3. *Consideriamo la rete di Petri in Figura 3.1, dove l'insieme di transizioni osservabili è* $T_o = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ *e l'insieme di transizioni non osservabili è* $T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{14}, \varepsilon_{15}\}$. *L'insieme delle transizioni di guasto è* $T_f = T_f^1 \cup T_f^2$, *dove* $T_f^1 = \{\varepsilon_{12}\}$ *e* $T_f^2 = \{\varepsilon_{14}\}$. *La funzione di etichettatura è* $\mathscr{L}(t_1) = a$, $\mathscr{L}(t_2) = \mathscr{L}(t_3) = b$, $\mathscr{L}(t_4) = \mathscr{L}(t_5) = c$, $\mathscr{L}(t_6) = \mathscr{L}(t_7) = d$. *Il MBRG corrispondente è mostrato in Figura 3.2. Dove* $M_0 = [1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0]^T$, $M_1 = [0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1]^T$, $M_2 = [0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0]^T$, $M_3 = [0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0]^T$, $M_4 = [0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0]^T$, $M_5 = [0\,0\,0\,0\,0\,0\,0\,1\ \ 0\,0\,0\,0]^T$, $M_6 = [0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0]^T$, $M_7 = [0\,0\,0\,0\,0\,0\,1\,0\,0\ \ 0\,0\,0]^T$ $e_1 = [0\,0\,0\,0\,0\,0]^T$, $e_2 = [1\,0\,0\,0\,0\,0]^T$, $e_3 = [0\,0\,0\,0\,1\,0]^T$, $e_4 = [0\,1\,1\,1\,0\,0]^T$, $e_5 = [0\,1\,0\,0\,0\,0]^T$, $e_6 = [0\,0\,0\,0\,0\,1]^T$.

Ogni nodo contiene una differente marcatura di base e un vettore di due elementi (perché ci sono due classi di guasto). Ad esempio, a M_2 è associato il vettore [1 0] *perché $\mathcal{T}(M_2)$ ammette soluzione per la prima classe di guasto, cioé dalla marcatura M_2 può verificarsi il guasto in T_f^1.*

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_{12}, e_5) da M_2. ■

3.1.5 Modified Basis Reachability Diagnoser (MBRD) vs Basis Reachability Diagnoser (BRD)

Nel lavoro [10] la diagnosticabilità veniva valutata utilizzando un diagnosticatore chiamato *Basis Reachability Diagnoser* (BRD). Si tratta di un grafo deterministico che usato in aggiunta all'MBRG permette di ricavare le condizioni necessarie e sufficienti per la diagnosticabilità. Il BRD viene definito come segue:

Definizione 3.1.9. *Il Diagnosticatore di Raggiungibilita' 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 L. ■

Il BRD può essere facilmente determinato a partire dal MBRG. In particolare, i valori di M ed x sono leggibili dal 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 $l \in L$ dobbiamo considerare solo le marcature raggiunte tramite lo scatto di $Y_{min}(M, t)$ per tutte le transizioni t tali che $l = \mathscr{L}(t)$.

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.

In questa tesi ci siamo occupati della costruzione del *Diagnosticatore di raggiungibilità di base modificato*, questo nuovo grafo consente di determinare in unione all'MBRG le condizioni necessarie e sufficienti per la diagnosticabilità, con un miglioramento dal punto di vista delle prestazioni rispetto all'approccio precedente.

Definizione 3.1.10. *Il MBRD è un grafo deterministico in cui ciascun nodo contiene le seguenti informazioni:*

- *Una o più coppie* (M, h)*, dove:*
  - M *è una marcatura;*
  - $h \in \{N, F\}^{|T_f|}$ *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;*
- r *etichette* Δ_i*, per* $i = 1, \ldots, r$*, che rappresentano lo stato di diagnosi del nodo in riferimento alle* r *classi di guasto.*

Gli archi sono etichettati con un simbolo in L. ■

Il nodo iniziale dell'MBRD comprende la marcatura iniziale dell'MBRG più tutte quelle marcature raggiunte a partire da questa con lo scatto di transizioni di guasto.

A partire da ciascuna marcatura in questo nodo verifichiamo quali etichette sono abilitate, presa una data etichetta il nodo successivo comprenderà tutte le marcature raggiunte con lo scatto dell'etichetta a partire da ogni marcatura del nodo precedente, più tutte quelle marcature che possono essere raggiunte con lo scatto di transizioni di guasto.

A differenza del BRD ogni nodo non è costituito dalle sole marcature di base ma dalle possibili marcature presenti nel MBRG.

Algoritmo 3.1.2. ***[Calcolo del MBRD]***

1. *Etichetta il nodo iniziale* $d_0 = (M_0, h_0)$, $d_0 = \mathbb{N}^r$.
1. $\forall \sigma \in T_f^*$: σ *è abilitata da* M_0, *i.e.,* $M_0[\sigma\rangle$
  2.1. *sia* (M, h) *la coppia in cui*
    M *è la marcatura raggiunta scattando* σ *da* M_0, *i.e.,* $M_0[\sigma\rangle M_0$, *e*
    $$h_i = \begin{cases} N & \text{if } \forall\ t_f \in T_f^i,\ t_f \notin \sigma, \\ F & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, r$$
  2.2. *se* $(M, h) \notin d_0$,
      sia $d_0 = d_0 \cup \{(M, h)\}$.
3. $\forall\ i = 1, \ldots, r$,
  3.1. *se* $\forall\ (M, h) \in d_0,\ h_i = N$,
      sia $\Delta_i = 0$,
  altrimenti $\forall\ (M, h) \in d_0,\ h_i = F$,
      sia $\Delta_i = 3$,
  altrimenti
      sia $\Delta_i = 2$.
4. *Non assegnare tag al nodo.*
5. *Fintanto che esistono nodi senza tag,*
  5.1. *seleziona un nodo* d *senza tag,*
  5.2. $\forall\ l \in L$,
  5.3. *sia* $d' = \emptyset$,
    5.3.1. $\forall M\ :\ (M, h) \in d$,
      5.3.1.1. $\forall \sigma \in T_f^*$: $M[l\sigma\rangle$,
        – *sia* (M', h') *la coppia in cui*
          M' : $M[l\sigma\rangle M'$ *e*

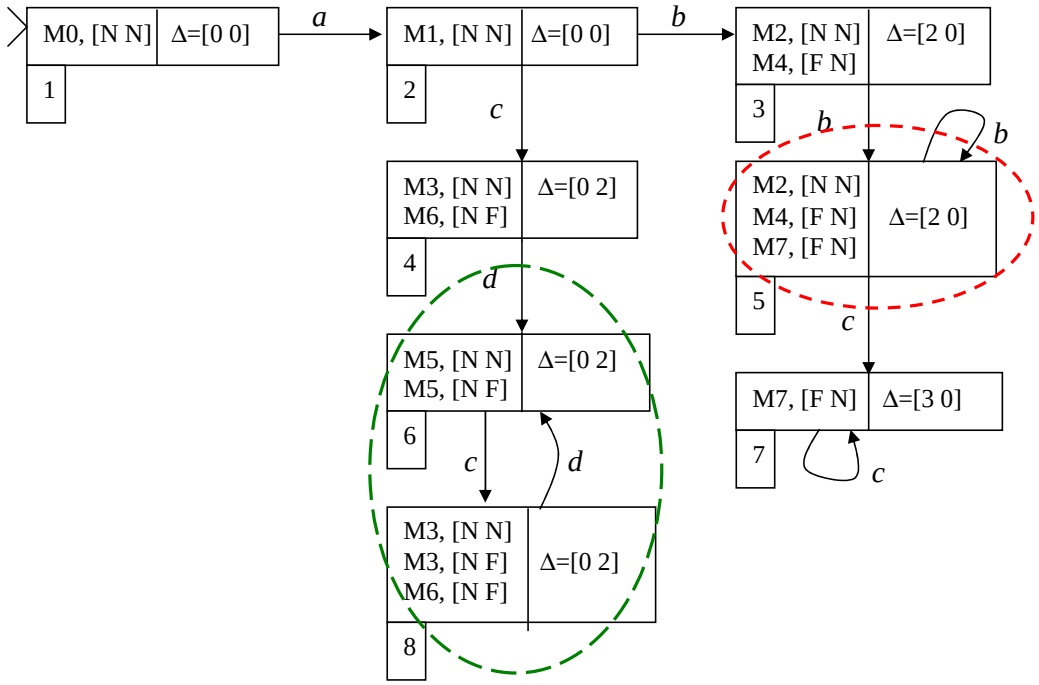


Figura 3.3: L'MBRD della RdP in Figura 3.1.

$$h'_i = \begin{cases} N & \text{if } h_i = N \text{ and } \forall\ t_f \in T_f^i,\ t_f \notin \sigma, \\ F & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, r$$

– *se* $(M', h') \notin d'$,
sia $d' = d' \cup \{(M', h')\}$.

5.3.2. $\forall\ i = 1, \ldots, r$,

5.3.2.1. *se* $\forall\ (M, h) \in d',\ h_i = N$,
sia $\Delta_i = 0$,
altrimenti $\forall\ (M, h) \in d',\ h_i = F$,
sia $\Delta_i = 3$,
altrimenti
sia $\Delta_i = 2$.

5.3.3. *Se* $\nexists$ *un nodo* $\bar{d} = d'$ *nel grafo,*
aggiungi un nuovo nodo d' *al grafo.*

5.3.4. *Aggiungi un arco* l *da* d *a* d'.

5.4. *Tagga* d *old.*

5.5. *Vai al passo 5.1.*

6. *Rimuovi tutti i tag.*

■

Esempio 3.1.4. *: In Figura 3.3 è riportato l'MBRD relativo alla RdP in Figura 3.1, con* $T_f = T_f^1 \cup T_f^2$*, dove* $T_f^1 = \{\varepsilon_{12}\}$ *e* $T_f^2 = \{\varepsilon_{14}\}$*. Il nodo iniziale* 1 *comprende in questo caso solo la marcatura iniziale* M_0 *con valore di* $\vec{h}$ *pari a* $[NN]$*, in quanto nel MBRG non ci sono transizioni di guasto che possono scattare a partire dalla marcatura* M_0*. Lo stato di diagnosi in questo caso è* $\Delta = [0\ 0]$*. Dal nodo iniziale* 1 *è abilitata l'etichetta* a *che conduce al nodo* 2*. Lo stato di diagnosi del nodo* 2 *è ancora* $\Delta = [0\ 0]$*. Dal nodo* 2 *sono abilitate le etichette* b *e* c *e conducono rispettivamente ai nodi* 3 *e* 4*. Lo stato di diagnosi del nodo* 3 *è* $\Delta = [2\ 0]$*, in quanto in tale nodo sono presenti sia marcature con valore del vettore* $\vec{h}$ *pari a N sia marcature con valore di* $\vec{h}$ *pari a F, per la prima classe di guasto; mentre per la seconda classe di guasto, le marcature presenti nel nodo hanno tutte valore del vettore* $\vec{h}$ *pari a N. Lo stato di diagnosi del nodo* 2 *è* $\Delta = [0\ 2]$*. E così via fino ad analizzare tutti i nodi del grafo.* ■

Proposizione 3.1.3. *Nel MBRD non esistono nodi con stato di diagnosi pari a* 1 *(*$\Delta = 1$*).* ■

Dimostrazione. Gli stati con $\Delta = 1$ sono quelli che contengono le marcature di base raggiunte senza scattare il guasto, ma da cui lo scatto del guasto è possibile. Per costruzione i nodi del MBRD contengono anche tutte le marcature raggiunte con lo scatto di transizioni di guasto, di conseguenza il caso con stato di diagnosi pari a 1 non è presente.

3.1.6 Condizioni necessarie e sufficienti per la diagnosticabilità

In questo paragrafo vengono fornite le condizioni necessarie e sufficienti per la diagnosticabilità basate sui concetti di cicli incerti e indeterminati. Queste condizioni possono essere verificate usando il MBRD in unione con il MBRG. La prima cosa da fare è verificare se il MBRD contiene un ciclo incerto, che risulta essere un potenziale ciclo indeterminato, e successivamente utilizzare il MBRG per verificare se tale ciclo è indeterminato oppure no.

Definizione 3.1.11. *: Sia* γ *un ciclo nel MBRD con proiezione osservabile* $\rho \in L^*$*. Il ciclo* γ *è incerto in riferimento alla classe di guasto* T_f^i *se e solo se include stati con* $\Delta_i = 2$*.* ■

Definizione 3.1.12. *Sia* γ *un ciclo incerto nel MBRD con proiezione osservabile* $\rho \in L^*$*. Il ciclo* γ *è indeterminato in riferimento alla classe di guasto* T_f^i *se nel MBRG esistono due cicli* γ_1 *e* γ_2 *che soddisfano le seguenti condizioni:*

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

(ii) γ_2 *contiene un guasto in* T_f^i*, mentre* γ_1 *non contiene un guasto in* T_f^i*.* . ■

In altre parole affinché un ciclo incerto nel MBRD sia indeterminato, deve esistere nel MBRG almeno un ciclo N, ossia un ciclo composto da nodi del MBRG con valore del vettore $\vec{h}$, nel nodo del MBRD, pari a N e almeno un ciclo F, ossia un ciclo costituito da nodi del MBRG con valore del vettore $\vec{h}$ nel nodo del MBRD, pari a F, relativamente alla classe di guasto in esame. Una volta individuato un ciclo incerto nel MBRD per la classe di guasto i-esima, per la determinazione dei cicli N e dei cicli F si procede in questo modo:

- CICLI N: per ogni nodo del ciclo incerto trovato nel MBRD si individuano al suo interno tutte le marcature (corrispondenti a nodi del MBRG) con valore del vettore $\vec{h}$ pari a N, per la classe di guasto in esame. A questo punto i cicli N vengono ottenuti come combinazioni dei nodi con $h = N$ individuati in ciascun nodo del ciclo incerto;
- CICLI F: per ogni nodo del ciclo incerto trovato nel MBRD si individuano al suo interno tutte le marcature (corrispondenti a nodi del MBRG) con valore del vettore $\vec{h}$ pari a F, per la classe di guasto in esame. A questo punto i cicli F vengono ottenuti come combinazioni dei nodi con $h = F$ individuati in ciascun nodo del ciclo incerto.

Esempio 3.1.5. *Consideriamo il MBRD in Figura 3.3 corrispondente alla RdP in Figura 3.1. L'ellisse tratteggiata in rosso rappresenta il ciclo incerto per la classe di guasto* T_f^1*, mentre l'ellisse tratteggiata in verde rappresenta il ciclo incerto per la classe di guasto* T_f^2*.*

Analizziamo la prima classe di guasto. Il ciclo incerto è $\gamma = 5\,\underrightarrow{b}\,5$ *per il quale* $\rho = b$*. Una volta individuato il ciclo incerto è necessario andare ad analizzare tutti i possibili cicli N e cicli F, per la classe di guasto in esame. L'unico ciclo N da verificare nel MBRG è:*

$$M_2\,\underrightarrow{b}\,M_2$$

mentre i cicli F da verificare sono:

$$M_4\,\underrightarrow{b}\,M_4$$

e

$$M_5\,\underrightarrow{b}\,M_5.$$

Andando ad analizzare il MBRG in Figura 3.2 possiamo osservare che il ciclo N, $M_2 \underrightarrow{b} M_2$*, è presente. A questo punto andiamo a verificare se almeno uno dei due cicli F è presente nel MBRG. Nessuno dei due cicli è presente, per cui possiamo concludere che il ciclo incerto nel MBRD,* $\gamma = 5 \underrightarrow{b} 5$*, non è indeterminato, di conseguenza la classe di guasto* T_f^1 *è diagnosticabile.*

Analizziamo ora la seconda classe di guasto. Il ciclo incerto presente nel MBRD è $\gamma = 6 \underrightarrow{c} 8 \underrightarrow{d} 6$*, con proiezione osservabile* $\rho = cd$*. L'unico ciclo N da verificare nel MBRG è:*

$$M_5 \underrightarrow{c} M_3 \underrightarrow{d} M_5$$

mentre i cicli F da verificare nel MBRG sono:

$$M_5 \underrightarrow{c} M_3 \underrightarrow{d} M_5$$

e

$$M_5 \underrightarrow{c} M_6 \underrightarrow{d} M_5$$

in questo caso è presente sia il ciclo N sia entrambe i cicli F. Il ciclo è indeterminato se viene individuato, nel MBRG, almeno un ciclo N e almeno un ciclo F. In questo caso il ciclo incerto individuato nel MBRD per la seconda classe di guasto è indeterminato, di conseguenza la classe di guasto T_f^2 *è non diagnosticabile.*

■

Teorema 3.1.1. *Un sistema di rete* $\langle N, M_0 \rangle$ *è diagnosticabile in riferimento alla classe di guasto* T_f^i *se e solo se il suo MBRD non presenta cicli indeterminati relativamente a* T_f^i*.* ■

Corollario 3.1.1. *Un sistema di rete* $\langle N, M_0 \rangle$ *è diagnosticabile se e solo se il suo MBRD non presenta cicli indeterminati rispetto a ogni classe di guasto.* ■

Capitolo 4

Algoritmo di Johnson per il calcolo dei cicli in un grafo

Abbiamo visto come la verifica della diagnosticabilità sia basata sui concetti di cicli incerti e indeterminati. Per la ricerca dei cicli incerti all'interno del MBRD si è fatto uso dell'algoritmo di Donald B. Johnson [18].

Si tratta di un algoritmo che trova tutti i cicli elementari di un grafo orientato. Un *ciclo* è un percorso in cui il primo e l'ultimo nodo sono identici. Un ciclo è *elementare* se nessun nodo appare due volte.

Il tempo di calcolo è limitato da

$$O((n+e)(c+1))$$

e lo spazio è limitato da

$$O(n+e)$$

dove n è il numero dei nodi del grafo, e è il numero di archi e c è il numero dei cicli elementari nel grafo [18]. L'algoritmo è simile agli algoritmi di Tiernan [28] e Tarjan [27].

Tale algoritmo esclude grafi con loop (archi della forma (v, v)) e archi multipli tra una coppia di vertici.

L'algoritmo di Johnson è fondamentalmente un algoritmo di ricerca in profondità (DFS - depth-first search) che è stato modificato per migliorare l'efficienza

della ricerca. Le etichette intere $\{1, 2, ..., n\}$ sono assegnate ai nodi del grafo e i cicli sono definiti iniziando dal nodo con etichetta minore. Ogni nodo s del grafo è scelto successivamente, e nel grafo vengono ricercati i cicli legati a quel nodo. Quando i cicli legati al nodo s sono stati trovati, viene scelto il nodo successivo $s+1$ e la ricerca nel grafo continua. L'algoritmo utilizza il concetto di componenti fortemente connesse. Le componenti fortemente connesse hanno la proprietà che tutti i nodi della componente sono mutuamente raggiungibili. Esse vengono utilizzate per individuare le zone del grafo che contengono tutti i cicli che hanno radice in un determinato nodo s. In particolare un grafo F è indotto dal grafo originale G dai nodi $\{s, s+1, ..., n\}$. Le componenti fortemente connesse hanno la proprietà che tutti i nodi sono raggiungibili dagli altri nodi. Allora la componente fortemente connessa contenente il nodo s contiene tutti i cicli che hanno radice in s.

Durante la ricerca per cicli con radice nel nodo s, vengono memorizzate le informazioni relative ai nodi e agli archi che non conducono a cicli. Questo impedisce che questi nodi e archi vengano cercati di nuovo in futuro. Questa informazione è memorizzata usando nodi bloccati e insiemi B. Fondamentalmente i nodi vengono bloccati quando vengono aggiunti al percorso di ricerca. Ogni nodo ha anche un B-set che contiene i suoi predecessori che dovrebbero venire sbloccati quando il nodo viene sbloccato.

I cicli elementari sono costruiti a partire da un vertice radice s nel sottografo indotto da s. Per evitare che ci siano cicli doppi, un vertice v è bloccato quando viene aggiunto al percorso elementare che inizia in s. Esso rimane bloccato fino a che ciascun percorso da v a s interseca il percorso elementare corrente in un vertice diverso da s. L'algoritmo accetta un grafo G rappresentato da una struttura di adiacenza A_G composta da una lista di adiacenza $A_G(v)$ per ogni $v \in V$. La lista $A_G(v)$ contiene u se è solo se l'arco $(v, u) \in E$. L'algoritmo assume che i vertici siano rappresentati da numeri interi da 1 a n. L'algoritmo procede con la costruzione dei percorsi elementari a partire dal vertice s. I vertici del percorso elementare corrente sono tenuti in uno stack. Un vertice viene accodato ad un percorso elementare tramite una chiamata alla procedura CIRCUIT e viene cancellato al ritorno da questa chiamata. Quando un vertice v viene aggiunto ad un percorso, esso è bloccato impostando *blocked(v) = true*, così che v non possa essere utilizzato due volte nello stesso percorso.

L'algoritmo in pseudo-codice è mostrato nella Figura 4.1.

```
    CIRCUIT-FINDING ALGORITHM
begin
    integer list array A_K(n), B(n); logical array blocked (n); integer s;
    logical procedure CIRCUIT (integer value v);
        begin logical f;
            procedure UNBLOCK (integer value u);
                begin
                    blocked (u) := false;
                    for w∈B(u) do
                        begin
                            delete w from B(u);
                            if blocked(w) then UNBLOCK(w);
                        end
                end UNBLOCK;
            f := false;
            stack v;
            blocked(v) := true;
L1:         for w∈A_K(v) do
                if w=s then
                    begin
                        output circuit composed of stack followed by s;
                        f := true;
                    end
                else if ¬blocked(w) then
                        if CIRCUIT(w) then f := true;
L2:         if f then UNBLOCK(v)
            else for w∈A_K(v) do
                        if v∉B(w) then put v on B(w);
            unstack v;
            CIRCUIT := f;
        end CIRCUIT;
    empty stack;
    s := 1;
    while s < n do
        begin
            A_K := adjacency structure of strong component K with least
                   vertex in subgraph of G induced by {s, s+1, · · · , n};
            if A_K ≠ ∅ then
                begin
                    s := least vertex in V_K;
                    for i∈V_K do
                        begin
                            blocked(i) := false;
                            B(i) := ∅;
                        end;
L3:                 dummy := CIRCUIT(s);
                    s := s+1;
                end
            else s := n;
        end
end;
```

Figura 4.1: Pseudo codice dell'Algoritmo di Johnson

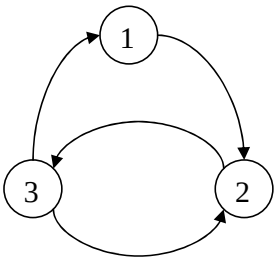


Figura 4.2: Esempio grafo per il calcolo dei cicli

4.1 Esempi di funzionamento dell'algoritmo

Esempio 4.1.1. *Supponiamo di applicare l'algoritmo di Johnson al grafo in Figura 4.2.*

In questo grafo è presente un'unica componente fortemente connessa composta dai nodi 1, 2 *e* 3*. I passaggi dell'algoritmo sono i seguenti:*

- $A_k = \begin{cases} 1 - - - > 2, \\ 2 - - - > 3, \\ 3 - - - > 1,2 \end{cases}$

  A_k è una struttura di adiacenza che mi dice quali sono i nodi della componente fortemente connessa in esame, in questo caso essa include i nodi 1, 2 *e* 3 *e per ognuno di questi nodi mi dice quali sono gli altri nodi che posso raggiungere.*

- $V_k = \{1,2,3\}$

  V_k è un vettore contenente i numeri dei nodi della componente fortemente connessa in esame.

  inizialmente la pila è vuota:

  $stack = \varepsilon$*;*

  il più piccolo vertice di V_k è $s = 1$;

  gli elementi del vettore blocked vengono tutti settati a "false", quindi questo significa che inizialmente tutti i nodi sono sbloccati.

  $blocked(1) = false$, $blocked(2) = false$, $blocked(3) = false$*;*

  gli insiemi B associati a ciascun nodo sono inizialmente vuoti.

  $B(1) = \varnothing$, $B(2) = \varnothing$, $B(3) = \varnothing$*;*

viene richiamata la funzione CIRCUIT(s) ossia CIRCUIT(1):

- *CIRCUIT(1)*

  $f = false;$

  il nodo 1 *viene inserito nella pila e viene bloccato:*

  $stack = [1];$

  $blocked = [true, false, false];$

  considero i nodi w appartenenti alla struttura di adiacenza del nodo 1*, in questo caso c'è solo* $w = 2$*:*

  $w = 2;$

  viene richiamata la funzione CIRCUIT(w) ossia CIRCUIT(2):

- *CIRCUIT(2)*

  $f = false;$

  il nodo 2 *viene inserito nella pila e viene bloccato:*

  $stack = [1\ 2];$

  $blocked = [true, true, false];$

  considero i nodi w appartenenti alla struttura di adiacenza del nodo 2*, in questo caso c'è solo* $w = 3$*:*

  $w = 3;$

  viene richiamata la funzione CIRCUIT(w) ossia CIRCUIT(3):

- *CIRCUIT(3)*

  $f = false;$

  il nodo 3 *viene inserito nella pila e viene bloccato:*

  $stack = [1\ 2\ 3];$

  $blocked = [true, true, true];$

  considero i nodi w appartenenti alla struttura di adiacenza del nodo 3*, in questo caso c'è* $w = 1$ *e* $w = 2$*:*

  $w = 1 = s;$

  siccome $w = 1$ *coincide con s, ossia con il nodo di partenza, abbiamo ottenuto un ciclo, lo mandiamo in uscita e poniamo la variabile f a true.*

  OUTPUT = 1 2 3 1 *primo ciclo trovato*

  $f = true;$

$w = 2$ *essendo il nodo* 2 *bloccato non succede nulla;*

a questo punto essendo f = *true possiamo sbloccare via via ciascun nodo, eliminarlo dalla pila e terminare tutte le chiamate alle funzioni CIRCUIT:*

$UNBLOCK(3)$ —> $blocked = [true, true, false]$;

$unstack\ 3$ —> $stack = [1\ 2]$;

$CIRCUIT(3) = f = true$;

$f = true$;

$UNBLOCK(2)$ —> $blocked = [true, false, false]$;

$unstack\ 2$ —> $stack = [1]$;

$CIRCUIT(2) = f = true$;

$f = true$;

$UNBLOCK(1)$ —> $blocked = [false, false, false]$;

$unstack\ 1$ —> $stack = \varepsilon$;

$CIRCUIT(1) = f = true$;

il passo successivo è quello di scegliere un nuovo nodo ossia $s = s + 1 = 2$ *e verificare se ci sono cicli legati a questo nodo.*

Non consideriamo più il nodo 1 *e gli archi entranti e uscenti da esso, per cui la matrice* A_k *corrispondente è la seguente:*

- $A_k = \begin{cases} 2 - - - > 3, \\ 3 - - - > 2 \end{cases}$
- $V_k = \{2, 3\}$;

  si procede come al passo precedente e si arriva così ad individuare un nuovo ciclo:

  $stack = \varepsilon$;

  $blocked(1) = false$, $blocked(2) = false$, $blocked(3) = false$;

  $B(1) = \varnothing$, $B(2) = \varnothing$, $B(3) = \varnothing$;

- $CIRCUIT(2)$

  $f = false$;

  $stack = [2]$;

$blocked = [false, true, false]$;

$w = 3$;

- *CIRCUIT(3)*

  $f = false$;

  $stack = [2\ 3]$;

  $blocked = [false, true, true]$;

  $w = 2 = s$;

  OUTPUT = 2 3 2 *secondo ciclo trovato*

  $f = true$;

  $UNBLOCK(3)$ —> $blocked = [false, true, false]$;

  $unstack\ 3$ —> $stack = [2]$;

  $CIRCUIT(3) = f = true$;

  $f = true$;

  $UNBLOCK(2)$ —> $blocked = [false, false, false]$;

  $unstack\ 2$ —> $stack = \varepsilon$;

  $CIRCUIT(2) = f = true$.

L'algoritmo termina in quanto viene applicato fin tanto che s è minore di n, ossia del numero di nodi.

Quindi i cicli trovati nel grafo sono: $[1 \to 2 \to 3 \to 1]$ e $[2 \to 3 \to 2]$.

Consideriamo un altro esempio di calcolo dei cicli.

Esempio 4.1.2. *Consideriamo il grafo in Figura 4.3. In tale grafo sono presenti tre componenti fortemente connesse, la prima comprende soltanto il nodo* 1*, la seconda comprende i nodi* 2 *e* 4 *mentre la terza comprende i nodi* 3*,* 5 *e* 6*. La prima componente essendo costituita da un solo nodo non viene considerata nel calcolo dei cicli.*

I passaggi dell'algoritmo sono i seguenti:

Viene analizzata la componente comprendente i nodi 2 *e* 4*.*

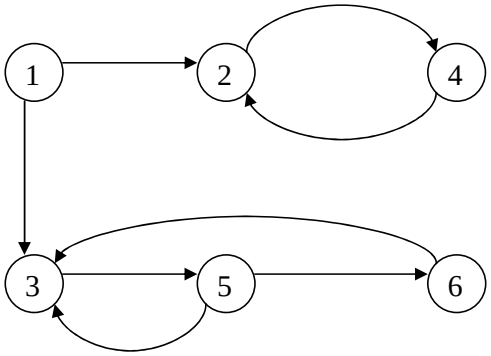


Figura 4.3: Esempio grafo per il calcolo dei cicli

- $A_k = \begin{cases} 2 - - - > 4, \\ 4 - - - > 2 \end{cases}$

  A_k è una struttura di adiacenza che mi dice quali sono i nodi della componente fortemente connessa in esame, in questo caso essa include i nodi 2, *e* 4 *e per ognuno di questi nodi mi dice quali sono gli altri nodi che posso raggiungere.*

- $V_k = \{2, 4\}$

  V_k è un vettore contenente i numeri dei nodi della componente fortemente connessa in esame.

  inizialmente la pila è vuota:

  $stack = \varepsilon$

  il più piccolo vertice di V_k è $s = 2$

  gli elementi del vettore blocked vengono tutti settati a "false", quindi questo significa che inizialmente tutti i nodi sono sbloccati.

  $blocked(1) = false$, $blocked(2) = false$, $blocked(3) = false$, $blocked(4) = false$, $blocked(5) = false$, $blocked(6) = false$;

  gli insiemi B associati a ciascun nodo sono inizialmente vuoti.

  $B(1) = \varnothing$, $B(2) = \varnothing$, $B(3) = \varnothing$, $B(4) = \varnothing$, $B(5) = \varnothing$, $B(6) = \varnothing$;

  viene richiamata la funzione CIRCUIT(s) ossia CIRCUIT(2):

- *CIRCUIT(2)*

  $f = false$;

  il nodo 2 *viene inserito nella pila e viene bloccato:*

  $stack = [2]$;

$blocked = [false, true, false, false, false, false]$;

considero i nodi w appartenenti alla struttura di adiacenza del nodo 1*, in questo caso c'è solo* $w = 4$*:*

$w = 4$;

viene richiamata la funzione CIRCUIT(w) ossia CIRCUIT(4):

- *CIRCUIT(4)*

  $f = false$;

  il nodo 4 *viene inserito nella pila e viene bloccato:*

  $stack = [2\ 4]$;

  $blocked = [false, true, false, true, false, false]$;

  considero i nodi w appartenenti alla struttura di adiacenza del nodo 4*, in questo caso c'è solo* $w = 2$*:*

  $w = 2 = s$

  siccome $w = 2$ *coincide con s, ossia con il nodo di partenza, abbiamo ottenuto un ciclo, lo mandiamo in uscita e poniamo la variabile f a true.*

  OUTPUT = 2 4 2 *primo ciclo trovato*

  $f = true$;

  a questo punto vengono sbloccati il nodo 4 e il nodo 2:

  $UNBLOCK(4)$ —> $blocked = [false, true, false, false, false, false]$;

  $unstack\ 4$ —> $stack = [2]$;

  $CIRCUIT(4) = f = true$;

  $f = true$;

  $UNBLOCK(2)$ —> $blocked = [false, false, false, false, false, false]$;

  $unstack\ 2$ —> $stack = \varepsilon$;

  $CIRCUIT(2) = f = true$;

  A questo punto viene analizzata la componente comprendente i nodi 3, 5 e 6.

- $A_k = \begin{cases} 3 - - - > 5, \\ 5 - - - > 6, 3, \\ 6 - - - > 3 \end{cases}$

- $V_k = \{3, 5, 6\}$

  inizialmente la pila è vuota:

  $stack = \varepsilon$

  il più piccolo vertice di V_k *è* $s = 3$

  gli elementi del vettore blocked vengono tutti settati a "false", quindi questo significa che inizialmente tutti i nodi sono sbloccati.

  $blocked(1) = false,\ blocked(2) = false,\ blocked(3) = false,$
  $blocked(4) = false,\ blocked(5) = false,\ blocked(6) = false;$

  gli insiemi B associati a ciascun nodo sono inizialmente vuoti.

  $B(1) = \varnothing,\ B(2) = \varnothing,\ B(3) = \varnothing,\ B(4) = \varnothing,\ B(5) = \varnothing,\ B(6) = \varnothing;$

  viene richiamata la funzione CIRCUIT(s) ossia CIRCUIT(3):

- *CIRCUIT(3)*

  $f = false;$

  il nodo 2 *viene inserito nella pila e viene bloccato:*

  $stack = [3];$

  $blocked = [false, false, true, false, false, false];$

  considero i nodi w appartenenti alla struttura di adiacenza del nodo 3*, in questo caso c'è solo* $w = 5$*:*

  $w = 5;$

  viene richiamata la funzione CIRCUIT(w) ossia CIRCUIT(5):

- *CIRCUIT(5)*

  $f = false;$

  il nodo 4 *viene inserito nella pila e viene bloccato:*

  $stack = [3\ 5];$

  $blocked = [false, false, true, false, true, false];$

  considero i nodi w appartenenti alla struttura di adiacenza del nodo 5*, in questo caso abbiamo* $w = 6$ *e* $w = 3$*:*

  $w = 6$

  viene richiamata la funzione CIRCUIT(w) ossia CIRCUIT(6):

- *CIRCUIT(6)*

  $f = false$;

  il nodo 4 *viene inserito nella pila e viene bloccato:*

  $stack$ = [3 5 6]*;*

  $blocked$ = [$false, false, true, false, true, true$]*;*

  considero i nodi w appartenenti alla struttura di adiacenza del nodo 6*, in questo caso abbiamo* $w = 3$*:*

  $w = 3 = s$

  siccome $w = 3$ *coincide con s, ossia con il nodo di partenza, abbiamo ottenuto un ciclo, lo mandiamo in uscita e poniamo la variabile f a true.*

  OUTPUT = 3 5 6 3 *primo ciclo trovato*

  $f = true$;

  a questo punto ho analizzato tutti i nodi che fanno parte della lista di adiacenza del nodo 6, quindi posso sbloccarlo:

  $UNBLOCK(6)$ —> $blocked$ = [$false, true, false, false, true, false$]*;*

  $unstack$ 6 —> $stack$ = [3 5]*;*

  $CIRCUIT(6) = f = true$;

  La chiamata alla funzione CIRCUIT(5) non è ancora terminata, bisogna ancora analizzare il nodo $w = 3$*:*

  $w = 3 = s$;

  siccome $w = 3$ *coincide con s, ossia con il nodo di partenza, abbiamo ottenuto un ciclo, lo mandiamo in uscita e poniamo la variabile f a true.*

  OUTPUT = 3 5 3 *secondo ciclo trovato* $f = true$;

  Sono stati esaminati tutti i nodi w appartenenti alla lista di adiacenza del nodo 5, per cui è possibili sbloccarlo:

  $UNBLOCK(5)$ —> $blocked$ = [$false, false, true, false, false, false$]*;*

  $unstack$ 5 —> $stack$ = [3]*;*

  $CIRCUIT(5) = f = true$;

  $f = true$;

  sblocchiamo anche il nodo 3:

  $UNBLOCK(3)$ —> $blocked$ = [$false, false, false, false, false, false$]*;*

  $unstack$ 3 —> $stack = \varepsilon$;

  $CIRCUIT(3) = f = true$;

Quindi i cicli trovati nel grafo sono: $[2 \rightarrow 4 \rightarrow 2]$ nella seconda componente e $[3 \rightarrow 5 \rightarrow 6 \rightarrow 3]$ e $[3 \rightarrow 5 \rightarrow 3]$ nella terza componente.

4.2 Studio della complessità computazionale del calcolo delle componenti fortemente connesse e dei cicli in un grafo

Abbiamo visto come la valutazione della diagnosticabilità di un sistema sia legata all'individuazione di certi cicli (cicli incerti) nel MBRD e nella verifica del fatto che essi siano indeterminati o no nel MBRG.

Il calcolo dei cicli è stato realizzato utilizzando l'algoritmo di Johnson che a sua volta utilizza al suo interno il concetto di componente fortemente connessa.

Ci è sembrato utile, a questo proposito, effettuare un'analisi sulla complessità computazionale del calcolo delle componenti fortemente connesse e dei cicli di un grafo. In questo modo è possibile fare delle considerazioni più accurate sull'efficienza dell'algoritmo per il calcolo dei cicli.

La teoria della complessità computazionale è una branca della teoria della computabilità che studia le risorse minime necessarie (principalmente tempo di calcolo e memoria) per la risoluzione di un problema.

I problemi vengono così classificati in differenti classi complessità, a seconda di quanto il migliore algoritmo di risoluzione noto sia efficiente.

Una distinzione informale ma di grande rilievo è quella posta tra i cosidetti problemi facili, di cui si conoscono algoritmi di risoluzione efficienti, e difficili, di cui gli unici algoritmi noti non sono efficienti. Con complessità di un algoritmo o efficienza di un algoritmo ci si riferisce alle risorse di calcolo da esso richieste.

Simulazioni numeriche

Per l'analisi della complessità computazionale sono state prese in considerazioni due reti di Petri parametriche, il parametro che viene fatto variare è il numero di gettoni, indicato con x, della marcatura iniziale.

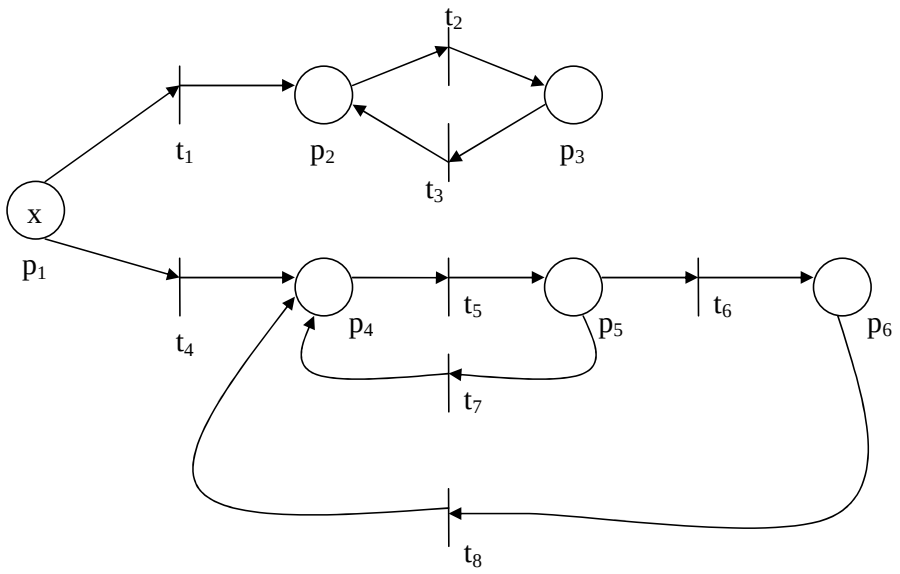


Figura 4.4: Esempio grafo per il calcolo dei cicli

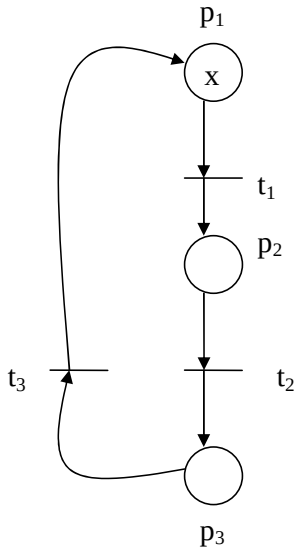


Figura 4.5: Esempio grafo per il calcolo dei cicli

Le due reti analizzate sono mostrate rispettivamente in Figura 4.4 e in Figura 4.5.

Per ogni valore di x, sono stati determinati:

- il grafo di raggiungibilità della rete (R);

- il numero di nodi del grafo (n_R);
- il numero di archi del grafo (a_R);
- il numero di componenti fortemente connesse presenti nel grafo (#cfc);
- la cardinalità di ciascuna componente (n_cfc);
- il numero di archi di ciascuna componente (a_cfc);
- il numero di cicli presenti in ciascuna componente (cicli_cfc);
- il tempo per la determinazione delle componenti fortemente connesse espresso in secondi (t_cfc);
- il tempo per il calcolo dei cicli espresso in secondi (t_cicli).

I risultati ottenuti dalle simulazioni sono mostrati nella Tabelle 4.1 e 4.2.

Le tabelle sono strutturate in questo modo:

- la colonna 1 indica il numero x dei gettoni presenti nella marcatura iniziale;
- le colonne 2, 3 e 4 indicano rispettivamente il numero di nodi, il numero di archi e il numero di componenti fortemente connesse del grafo di raggiungibilità della rete;
- le colonne 5, 6 e 7 indicano rispettivamente il numero di nodi, il numero di archi e il numero di cicli di ciascuna componente fortemente connessa individuata;
- le colonne 8 e 9 indicano il tempo impiegato per il calcolo delle componenti fortemente connesse e dei cicli, espresso in secondi.

Analisi dei risultati

Consideriamo la Tabella 4.1 relativa alla rete di Figura 4.4, in questo caso il numero x di gettoni della marcatura iniziale viene fatto variare da 1 a 4. La prima cosa che si può osservare è che, naturalmente, all'aumentare di x aumentano conseguentemente il numero di nodi del grafo di raggiungibilità, il numero di archi e il numero di componenti fortemente connesse.

x	n_R	a_R	$\#cfc$	n_cfc	a_cfc	$cicli\_cfc$	$t_{cfc}[s]$	$t_{cicli}[s]$
1	6	8	3	1-2-3	0-2-4	0-1-2	0.0011	0.0020
2	21	48	6	1-2-3-3-6-6	0-2-4-4-14-12	0-1-2-2-14-9	0.0022	0.0174
3	56	168	10	1-2-3-3-6-6-4-9-12-10	0-2-4-4-14-12-6-24-36-24	0-1-2-2-15-9-3-40-214-32	0.0240	0.3706
4	126	448	15	1-2-3-3-6-6-4-9-12-10-5-12-18-20-25	0-2-4-4-14-12-6-24-36-24-8-34-60-68-40	0-1-2-2-15-9-3-40-214-32-4-101-4160-12309-156	0.5197	36.0127

Tabella 4.1: Analisi complessità computazionale della rete in Fig. 4.4.

x	n_R	a_R	$\#cfc$	n_cfc	a_cfc	$cicli\_cfc$	$t_{cfc}[s]$	$t_{cicli}[s]$
1	3	3	1	3	3	1	0.0046	0.0026
2	6	9	1	6	9	5	0.0008	0.0047
3	10	18	1	10	18	16	0.0009	0.0361
4	15	30	1	15	30	58	0.0012	0.1338
5	21	45	1	21	45	304	0.0018	0.8020
6	28	63	1	28	63	2421	0.0029	8.6494

Tabella 4.2: Analisi complessità computazionale della rete in Fig. 4.5.

Analizzando le componenti fortemente connesse, è possibile osservare che al crescere di x, si presentano delle componenti con un numero di nodi e archi via via crescente. Il numero dei cicli individuati subisce una rapida crescita con l'aumentare della dimensione delle componenti fortemente connesse. Ad esempio, per $x = 1$ si ha che la terza componente fortemente connessa ha un numero di nodi pari a 3, un numero di archi pari a 4 e un numero di cicli pari a 2. Se andiamo ad analizzare il caso co $x = 3$, si può vedere come la nona componente fortemente connessa ha un numero di nodi pari a 12, un numero di archi pari a 36 e il numero di cicli è pari a 214.

Per quanto riguarda i tempi di calcolo, sia nel caso del calcolo delle componenti fortemente connesse sia in quello del calcolo dei cicli il tempo impiegato si moltiplica per 10 ad ogni incremento di x di una unità. Fatta eccezione per il tempo di calcolo delle componenti fortemente connesse nel caso di $x = 2$, in cui il tempo di calcolo rispetto al caso con $x = 1$ è raddoppiato. Per un valore di x pari a 4 il tempo per il calcolo delle componenti fortemente connesse è ancora dell'ordine di 10^{-1} mentre il tempo per il calcolo dei cicli è già dell'ordine di 10.

Consideriamo la Tabella 4.2 relativa alla rete in Figura 4.5, in questo caso, la rete considerata è tale che il grafo di raggiungibilità (per ogni valore di x) è fortemente connesso per cui il numero di componenti fortemente connesse individuate è sempre pari a uno. Anche in questo caso il numero di cicli cresce rapidamente ad ogni incremento di x di una unità. Ad esempio per $x = 1$ la componente fortemente connessa ha un numero di nodi pari a 3, un numero di archi pari a 3 e un numero di cicli pari a 1, mentre per $x = 6$ la componente fortemente connessa ha un numero di noci pari a 28, un numero di archi pari a 63 e un numero di cicli addirittura pari a 2421. Per quanto riguarda i tempi di calcolo dei cicli, nel passaggio da $x = 1$ a $x = 2$ il tempo di calcolo è più o meno raddoppiato, mentre negli altri casi il tempo aumenta di un ordine di 10 all'aumentare di x.

Capitolo 5

Toolbox di Matlab

Sommario

In questo capitolo, verranno illustrati i programmi che costituiscono il toolbox di Matlab, con alcuni esempi che ne chiariscano meglio il funzionamento. I programmi principali contenuti nel pacchetto riguardano la costruzione del *Modified Basis Reachability Diagnoser* (MBRD) a partire dal *Modified Basis Reachability Graph* (MBRG) (l'algoritmo per il calcolo dell'MBRG è stato fornito da Cabasino, Giua e Seatzu in [10]) e la verifica delle condizioni necessarie e sufficienti per la diagnosticabilità.

5.1 Introduzione ai programmi

Come già detto il toolbox implementato in questa tesi è un toolbox per Matlab. Trattandosi di funzioni Matlab esse verranno richiamate con la seguente sintassi:

[outpu1, output2,. . .] = nomefunzione[input1, input2,. . .]

la sintassi precedente sta a significare che: viene richiamata la funzione *nomefunzione* che prende in ingresso input1, input2, etc e restituisce in uscita output1, output2, etc.

Per aggiungere nuove funzioni è sufficiente creare un file di testo detto M-file. I file Matlab possono contenere un semplice insieme di istruzioni, oppure essere delle funzioni a se stanti caratterizzate da proprie variabili locali.

5.2 Programma MBRD.m

Questa funzione calcola il Modified Basis Reachability Diagnoser (MBRD). Viene richiamata in questo modo:

[D,numTclas,sizeE,no,nf]= MBRD(T,Pre,Post,M0,F,L,E)

Gli ingressi sono:

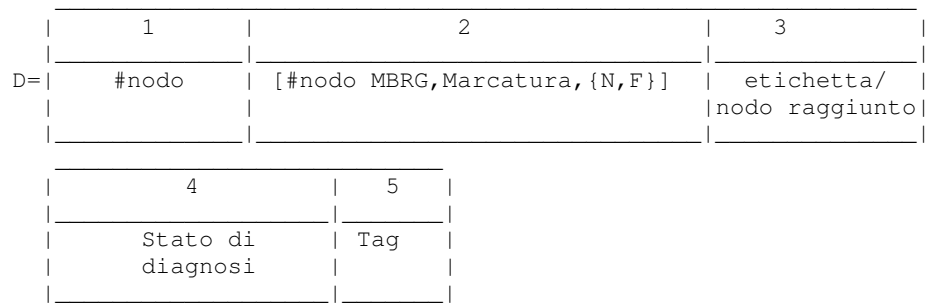
- T: è la matrice di celle restituita dalla funzione MBRG.m;
- Pre: matrice di pre-incidenza della rete;
- Post: matrice di post-incidenza della rete;
- M0: marcatura iniziale della rete;
- F: matrice di celle che definisce le classi di guasto. Essa ha dimensione [numero di classi di guasti × 1], tale che la i-esima riga contenga un vettore degli indici delle transizioni non osservabili che appartengono alla classe di guasti *i*;
- L: matrice di celle che definisce la funzione di etichettatura. Essa ha dimensione [numero di etichette × 1], tale che la i-esima riga contenga un vettore degli indici della transizioni osservabili corrispondenti all'etichetta i-esima;
- E: matrice di celle che definisce l'alfabeto delle etichette. Essa ha dimensione [numero di etichette × 1], tale che nella i-esima riga ci sia il simbolo associato alle transizioni osservabili che corrispondono all'etichetta i-esima.

Le uscite sono:

- D: è l'MBRD ed è rappresentato da una matrice di celle;

- numTclas: è il numero di classi di guasto;
- sizeE: è il numero di etichette;
- no: è il numero delle transizioni osservabili;
- nf: è il numero delle transizioni di guasto.

La matrice di celle D è fatta in questo modo:



La matrice D ha tante righe quanti sono i nodi del grafo corrispondente, ed ogni riga è costituita da 5 celle.

Cella 1: indica il numero del nodo dell'MBRD.

Cella 2: ha tante righe quante sono le marcature che fanno parte del nodo, il numero di colonne è pari a (1+p+numTclas) dove p è il numero di posti della rete e numTclas è il numero di classi di guasto. Il primo elemento di ciascuna riga indica il numero del nodo dell'MBRG corrispondente a quella marcatura.

Cella 3: mi da informazioni sulle etichette che possono scattare da quel nodo e sul numero del nodo che raggiungo.

Cella 4: indica lo stato di diagnosi. Questa informazione è contenuta in un vettore riga che ha un numero di elementi pari al numero di classi di guasto. Lo stato di diagnosi può assumere i seguenti valori: 0 (non si è verificato guasto), 2 (stato incerto), 3 (si è verificato il guasto).

Cella 5: indica un tag pari a 0 se il nodo non è stato ancora esplorato, 1 se il nodo è esplorato.

Il punto di partenza è la costruzione del nodo iniziale. Il primo nodo dell'MBRD comprende la marcatura iniziale M0 più tutte le marcature che vengono raggiun-

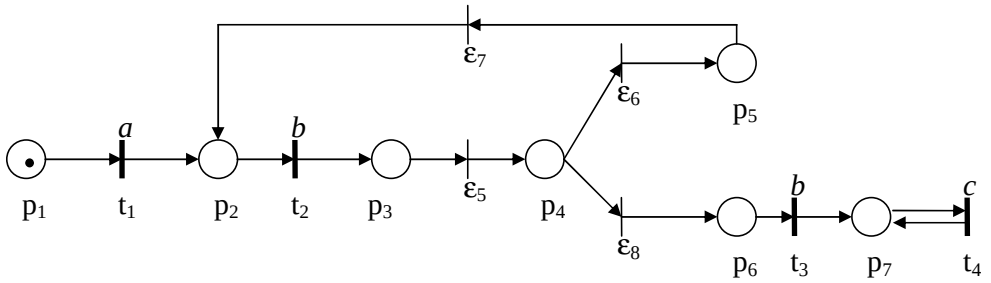


Figura 5.1: Esempio di RdP per il calcolo dell'MBRD

te a partire da questa con lo scatto di transizioni di guasto. Una volta costruito il primo nodo, esso viene esplorato. A partire da ciascuna marcatura in questo nodo verifico quali etichette sono abilitate, per ogni etichetta abilitata costruisco un nuovo nodo che comprenderà tutte le marcature raggiunte con lo scatto di quell'etichetta più tutte le marcature raggiunte con lo scatto di transizioni di guasto. Si procede in questo modo finchè tutti in nodi non sono stati esplorati (ossia Tag=1).

Presentiamo ora un esempio che mostri il funzionamento della precedente funzione.

Esempio 5.2.1. *: Consideriamo la RdP in Figura 5.1 in cui l'insieme delle transizioni osservabili è* $T_o = \{t_1, t_2, t_3, t_4\}$*,l'insieme delle transizioni non osservabili è* $T_u = \{\varepsilon_5, \varepsilon_6, \varepsilon_7, \varepsilon_8\}$*, le classi di guasto sono tre ,ossia* $T_f^1 = \{\varepsilon_5\}$*,* $T_f^2 = \{\varepsilon_7\}$ *e* $T_f^3 = \{\varepsilon_8\}$*. La funzione di etichettatura è:* $\mathscr{L}(t_1) = a, \mathscr{L}(t_2) = \mathscr{L}(t_3) = b$ *e* $\mathscr{L}(t_4) = c$*.*

Gli ingressi della funzione sono:

- *la matrice T in uscita dalla funzione MBRG.m;*
- *le matrici:*

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

- *la marcatura iniziale:*

$$M_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T;$$

- *gli array di celle:*

$$F = \left\{ \begin{array}{c} [5] \\ [7] \\ [8] \end{array} \right\}, \quad L = \left\{ \begin{array}{c} [1] \\ [2,\ 3] \\ [4] \end{array} \right\}, \quad E = \left\{ \begin{array}{c} [a] \\ [b] \\ [c] \end{array} \right\}$$

A questo punto è necessario richiamare la funzione MBRD.m, l'output della funzione è mostrato in Figura 5.2. ■

5.3 Programma showMBRD.m

Abbiamo visto come l'MBRD sia dato da una matrice di celle, essa può risultare di difficile interpretazione, per cui per rendere più semplice la costruzione del grafo corrispondente è stata creata una funzione, chiamata showMBRD.m che estrapola le informazioni e le rende visibili all'utente.

La funzione viene richiamata in questo modo:

show_MBRD(D,numTclas)

Gli ingressi sono:

- la matrice di celle D;
- il numero di classi di guasto.

L'uscita è la visualizzazione testuale delle informazioni necessarie per la costruzione dell' MBRD.

Richiamando la funzione precedente per la visualizzazione del MBRD dell'Esempio 5.2.1 otteniamo il risultato mostrato in Figura 5.3. Il MBRD corrispondente è mostrato in Figura 5.4.

```
>> Pre=[1 0 0 0 0 0 0 0;0 1 0 0 0 0 0 0;0 0 0 0 1 0 0 0;
   0 0 0 0 0 1 0 1;0 0 0 0 0 0 1 0;0 0 1 0 0 0 0 0;0 0 0 1 0 0 0 0];
>> Post=[0 0 0 0 0 0 0 0;1 0 0 0 0 0 1 0;0 1 0 0 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 0 1;0 0 1 1 0 0 0 0];
>> M0=[1 0 0 0 0 0 0]';
>> F={[5];[7];[8]};
>> L={[1];[2,3];[4]};
>> E={['a'];['b'];['c']};
>> T = MBRG(Pre,Post,M0,F,L,E);
>> [D,numTclas,sizeE,no,nf]=MBRD(T,Pre,Post,M0,F,L,E)

D =

    [1]    [1x11 double]    {1x1 cell}    [1x3 double]    [1]
    [2]    [1x11 double]    {1x2 cell}    [1x3 double]    [1]
    [3]    [4x11 double]    {1x2 cell}    [1x3 double]    [1]
    [4]    [5x11 double]    {1x3 cell}    [1x3 double]    [1]
    [5]    [5x11 double]    {1x3 cell}    [1x3 double]    [1]
    [6]    [1x11 double]    {1x3 cell}    [1x3 double]    [1]
    [7]    [1x11 double]    {1x3 cell}    [1x3 double]    [1]


numTclas =

     3


sizeE =

     3


no =

     4


nf =

     3
```

Figura 5.2: Command Window di Matlab per la funzione MBRD.m relativa all'Esempio 5.2.1

```
>> showMBRD(D,numTclas)
Il numero di nodi del Modified Basis Reachability Diagnoser è: 7

 # Nodo N1  Delta = [0 0 0]
 Le marcature appartenenti al nodo e i corrispondenti vettori h sono:
 Marcatura M0 = [1 0 0 0 0 0 0]    h = [N N N]
 Transizioni osservabili abilitate a scattare:
 a ---> N2
                  *********
 # Nodo N2  Delta = [0 0 0]
 Le marcature appartenenti al nodo e i corrispondenti vettori h sono:
 Marcatura M1 = [0 1 0 0 0 0 0]    h = [N N N]
 Transizioni osservabili abilitate a scattare:
 b ---> N3
                  *********
 # Nodo N3  Delta = [2 2 2]
 Le marcature appartenenti al nodo e i corrispondenti vettori h sono:
 Marcatura M2 = [0 0 1 0 0 0 0]    h = [N N N]
 Marcatura M3 = [0 0 0 1 0 0 0]    h = [F N N]
 Marcatura M1 = [0 1 0 0 0 0 0]    h = [F F N]
 Marcatura M4 = [0 0 0 0 0 1 0]    h = [F N F]
 Transizioni osservabili abilitate a scattare:
 b ---> N4
                  *********
 # Nodo N4  Delta = [3 2 2]
 Le marcature appartenenti al nodo e i corrispondenti vettori h sono:
 Marcatura M2 = [0 0 1 0 0 0 0]    h = [F F N]
 Marcatura M5 = [0 0 0 0 0 0 1]    h = [F N F]
 Marcatura M3 = [0 0 0 1 0 0 0]    h = [F F N]
 Marcatura M1 = [0 1 0 0 0 0 0]    h = [F F N]
 Marcatura M4 = [0 0 0 0 0 1 0]    h = [F F F]
 Transizioni osservabili abilitate a scattare:
 b ---> N5
 c ---> N6
                  *********
 # Nodo N5  Delta = [3 3 2]
 Le marcature appartenenti al nodo e i corrispondenti vettori h sono:
 Marcatura M2 = [0 0 1 0 0 0 0]    h = [F F N]
 Marcatura M5 = [0 0 0 0 0 0 1]    h = [F F F]
 Marcatura M3 = [0 0 0 1 0 0 0]    h = [F F N]
 Marcatura M1 = [0 1 0 0 0 0 0]    h = [F F N]
 Marcatura M4 = [0 0 0 0 0 1 0]    h = [F F F]
 Transizioni osservabili abilitate a scattare:
 b ---> N5
 c ---> N7
                  *********
 # Nodo N6  Delta = [3 0 3]
 Le marcature appartenenti al nodo e i corrispondenti vettori h sono:
 Marcatura M5 = [0 0 0 0 0 0 1]    h = [F N F]
 Transizioni osservabili abilitate a scattare:
 c ---> N6
                  *********
 # Nodo N7  Delta = [3 3 3]
 Le marcature appartenenti al nodo e i corrispondenti vettori h sono:
 Marcatura M5 = [0 0 0 0 0 0 1]    h = [F F F]
 Transizioni osservabili abilitate a scattare:
 c ---> N7
                  *********
```

Figura 5.3: Command Window di Matlab per la funzione showMBRD.m relativa all'Esempio 5.2.1

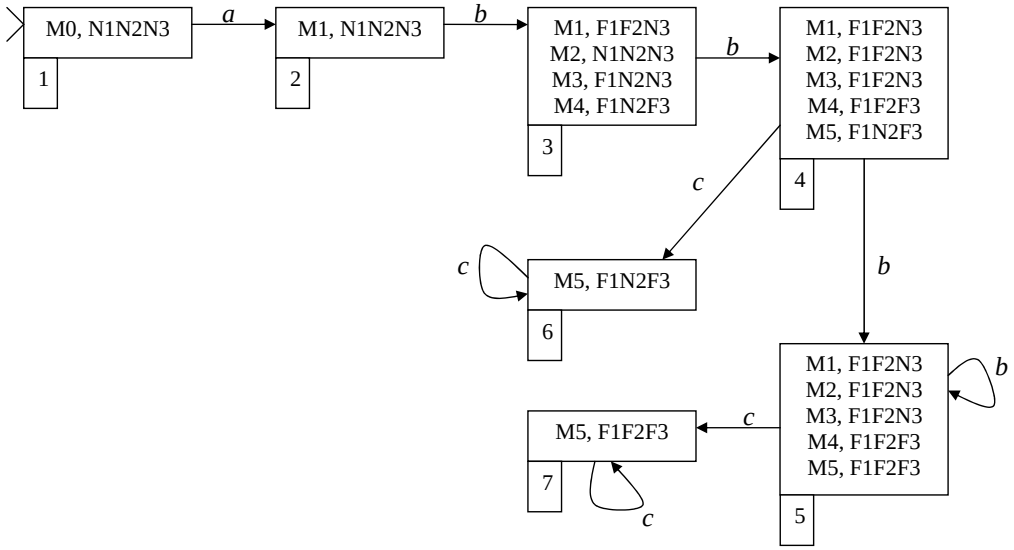


Figura 5.4: MBRD relativo alla RdP dell'Esempio 5.2.1

Una volta costruito l'MBRD per una data RdP è necessario stabilire se il sistema è diagnosticabile oppure no. Per la verifica della diagnosticabilità sono state costruite due funzioni, una di default che non appena determina che un dato ciclo per la classe di guasto i-esima è indeterminato,si interrompe e stabilisce che il sistema è non diagnosticabile per quella classe di guasto, ed una completa che invece analizza tutti i cicli incerti per ogni classe di guasto.

5.4 Programmi MBRD_diag_ott.m e MBRD_diag.m

La funzione *MBRD_diag_ott.m* è la versione di default mentre la funzione *MBRD_diag.m* è la versione completa.

Le funzioni vengono richiamate in questo modo:

MBRD_diag_ott(D,MBRG,sizeE,no,nf)

e

MBRD_diag(D,MBRG,sizeE,no,nf)

Gli ingressi sono:

```
>> MBRD_diag_ott(D,T,sizeE,no,nf)

********** CLASSE DI GUASTO 1 **********

Il sistema è diagnosticabile in riferimento alla classe di guasto 1
in quanto non sono presenti cicli incerti nel MBRD

********** CLASSE DI GUASTO 2 **********

Il sistema è diagnosticabile in riferimento alla classe di guasto 2
in quanto non sono presenti cicli incerti nel MBRD

********** CLASSE DI GUASTO 3 **********

Il sistema è diagnosticabile in riferimento alla classe di guasto 3
in quanto non sono presenti cicli indeterminati

************************************
Il sistema è diagnosticabile
************************************

Elapsed time is 0.005446 seconds.
```

Figura 5.5: Command Window di Matlab per la funzione MBRD_diag_ott.m relativa all'Esempio 5.2.1

- D, è la matrice di celle che rappresenta l'MBRD;
- il MBRG;
- sizeE, è il numero di etichette;
- no, è il numero di transizioni osservabili;
- nf, è il numero di transizioni di guasto.

L'uscita è la visualizzazione testuale delle informazioni relative alla diagnosticabilità: nel caso della versione di default ci viene detto semplicemente, per ciascuna classe di guasto, se il sistema è diagnosticabile o meno, nel caso della versione completa vengono fornite ulteriori informazioni, relative agli eventuali cicli incerti trovati nel MBRD, alla parola associata a tali cicli, ai possibili cicli N e cicli F e ai cicli N e F trovati nel MBRG.

Esempio 5.4.1. *Consideriamo l'MBRD in Figura 5.4 e richiamiamo per esso entrambe le funzioni per la verifica della diagnosticabilità. In Figura 5.5 è riportata l'uscita della funzione MBRD_diag_ott.m, mentre in Figura 5.6 è riportata l'uscita della funzione MBRD_diag.m.*

```
>> MBRD_diag(D,T,sizeE,no,nf)

********** CLASSE DI GUASTO 1 **********

 Il sistema è diagnosticabile in riferimento alla classe di guasto 1
in quanto non sono presenti cicli incerti nel MBRD

********** CLASSE DI GUASTO 2 **********

 Il sistema è diagnosticabile in riferimento alla classe di guasto 2
in quanto non sono presenti cicli incerti nel MBRD

********** CLASSE DI GUASTO 3 **********

 I cicli incerti trovati nel MBRD sono:
ans =
     5     5

La parola associata al ciclo
ans =
     5     5
è:
ans =
b

cicli_N =
     2     2
     3     3
     4     4

cicli_F =
     5     5
     6     6

cicli_F_trovati =
     []

cicli_N_trovati =
     2     2
     3     3
     4     4

Il sistema è diagnosticabile in riferimento alla classe di guasto 3
in quanto non sono presenti cicli indeterminati

************************************
Il sistema è diagnosticabile
************************************
Elapsed time is 0.056399 seconds.
```

Figura 5.6: Command Window di Matlab per la funzione MBRD_diag.m relativa all'Esempio 5.2.1

All'interno delle funzioni per la verifica della diagnosticabilità vengono richiamate ulteriori funzioni che descriviamo brevemente di seguito. Abbiamo già detto che la prima cosa da fare per la verifica della diagnosticabilità è la determinazione dei cicli incerti nell'MBRD (quelli formati da nodi con $\Delta = 2$).

Per ogni classe di guasto:

1. "disattivo" nel grafo i nodi con $\Delta \neq 2$ per la classe di guasto in esame, tramite la funzione MBRD_mod. All'interno di questa funzione viene richiamata la funzione MBRD_graph.m che estrapola le informazioni dalla matrice di celle restituita dalla funzione MBRD.m per ricavare il grafo corrispondente all'MBRD;
2. dal grafo ottenuto con la funzione MBRD_mod.m elimino i self loop e gli eventuali archi multipli tra due nodi (questo perché per il calcolo dei cicli incerti si utilizza l'algoritmo di Johnson che lavora con grafi che non hanno self loop e archi multipli tra due nodi), tramite la funzione elimina.m;
3. del grafo ottenuto con la funzione elimina.m calcolo le componenti fortemente connesse, tramite la funzione scc.m (funzione già esistente, presente nel pacchetto Automi);
4. richiamo la funzione cfc.m: questa funzione è stata creata per rendere più leggibile l'uscita della funzione scc.m;
5. calcolo i cicli tramite la funzione algoritmo_Johnson.m: essa richiama al suo interno la funzione CIRCUIT.m e questa funzione richiama al suo interno se stessa ricorsivamente e la funzione UNBLOCK.m.

5.4.1 Programma MBRD_graph.m

La funzione MBRD_graph.m trasforma la matrice di celle restituita dalla funzione MBRD.m,nella struttura dati necessaria per la funzione scc.m che calcola le componenti fortemente connesse di un grafo. Questo programma, a partire dall'MBRD mi costruisce un grafo con la seguente struttura:

```
      #nodo              label-->nodo        label-->nodo
G=      1    0    0        1     2             2       3
        2    0    0        1     1             2       2
        3    0    0        0     0             2       1
        1    2    1        0     0             0       0
```

La matrice G ha un numero di righe pari al numero dei nodi dell'MBRD più una. Nell'ultima riga, il secondo elemento indica il numero di etichette, mentre il terzo elemento è un flag (0 per l'albero e 1 per il grafo). La funzione viene richiamata in questo:

[G]=MBRD_graph(D,sizeE)

Gli ingressi sono:

- la matrice di celle D;
- il numero di etichette, sizeE.

L'uscita è il grafo G.

Questa funzione viene richiamata all'interno della funzione MBRD_mod.m.

5.4.2 Programma MBRD_mod.m

La funzione MBRD_mod.m modifica la matrice di celle D relativa all'MBRD disattivando per ciascuna classe di guasto i nodi che hanno Δdiverso da2 e i relativi archi uscenti ed entranti e costruendo una nuova matrice di celle MD. Tramite la funzione MBRD_graph.m viene ricavato il grafo corrispondente a questa matrice. La funzione viene richiamata in questo modo:

[MD,G]=MBRD_mod(D,sizeE,f)

Gli ingressi sono:

- la matrice di celle D, restituita dalla funzione MBRD.m;
- il numero di etichette indicato con sizeE;
- la classe di guasto in esame indicata con f.

Le uscite sono:

- la nuova matrice di celle MD;

- il grafo G corrispondente alla matrice MD.

5.4.3 Programma scc.m

La funzione scc.m calcola le componenti fortemente connesse di un grafo. Viene richiamata in questo modo:

comp=scc(G)

L'ingresso è il grafo G.

L'uscita è un vettore riga 'comp' che ha tanti elementi quanti sono i nodi. I nodi appartenenti alla stessa componente fortemente connessa hanno lo stesso numero identificativo. Infine l'eventuale segno negativo indica che la componente fortemente connessa è transitoria e non ergodica.

5.4.4 Programma cfc.m

Per poter rendere più semplice l'individuazione delle componenti fortemente connesse è stata scritta la funzione cfc.m. Viene richiamata in questo modo:

[CFC,Ak]=cfc(comp,G)

Gli ingressi sono:

- il vettore comp in uscita dalla funzione scc.m;
- il grafo G.

Le uscite sono:

- CFC: matrice di celle contenente in ogni riga una componente fortemente connessa;

- Ak: matrice di celle, costituita da tante righe quante sono le componenti fortemente connesse e per ciascun nodo di ciascuna componente mi dice quali sono i nodi che posso raggiungere (una sorta di matrice delle adiacenze).

5.4.5 Programma algoritmo_Johnson.m

Questa funzione calcola i cicli all'interno di un grafo orientato. Il calcolo dei cicli è basato sull'Algoritmo di Johnson ([18]). Si tratta di un algoritmo ricorsivo infatti, esso richiama al suo interno la funzione CIRCUIT, a sua volta la funzione CIRCUIT richiama al suo interno se stessa ricorsivamente e la funzione UNBLOCK. Anche la funzione UNBLOCK richiama se stessa ricorsivamente. La funzione viene richiamata in questo modo:

[CICLI]=algoritmo_Johnson(G,Ak,loop)

Gli ingressi sono:

- il grafo G;
- la matrice di celle Ak, restituita dalla funzione cfc.m;
- la matrice loop, restituita dalla funzione elimina.m.

L'uscita è la matrice di celle CICLI, contenente tutti i cicli elementari trovati più gli eventuali loop.

Vediamo ora un esempio che mostri il funzionamento di tale algoritmo.

Esempio 5.4.2. *Consideriamo il grafo in Figura 5.7, in questo grafo possiamo individuare due componenti fortemente connesse, la prima comprendente i nodi* 2 *e* 4 *e la seconda i nodi* 3, 5 *e* 6*. Il risultato ottenuto applicando l'algoritmo al grafo precedente è mostrato in Figura 5.8:*

Cerchiamo di commentare il risultato mostrato in Figura 5.8.

Il vettore "comp" restituito dalla funzione scc.m ha sei elementi tanti quanti sono i nodi del grafo in Figura 5.7.

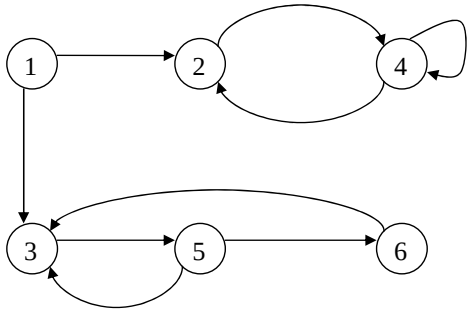


Figura 5.7: Grafo di esempio per l'algoritmo di Johnson

Il numero di componenti fortemente connesse è pari a 3 che non è altro che il massimo valore in modulo degli elementi del vettore "comp". La prima componente fortemente connessa comprende in nodo 1 ed è un componente transitoria, visto il segno negativo dell'identificativo numerico. La seconda componente fortemente connessa comprende i nodi 2 e 4 ed è una componente ergodica. La terza componente fortemente connessa comprende i nodi 3, 5 e 6 ed è anch'essa una componente ergodica.

La funzione elimina.m restituisce il grafo G privato dei self loop e degli archi multipli tra due nodi e una matrice chiamata "loop" che tiene traccia degli eventuali self loop trovati, in modo tale da poterli aggiungere ai cicli che verranno ricavati tramite la funzione algoritmo_Johnson.

La funzione "algoritmo_Johnson" restituisce una matrice di celle denominata CICLI che contiene tante celle quanti con i cicli trovati, nell'esempio che stiamo considerando possiamo osservare che sono stati trovati quattro cicli: il ciclo $2-4-2$, il ciclo $3-5-6-3$, il ciclo $3-5-3$ e un self loop $4-4$.

```
>> G=[1 0 0 1 2 2 3;2 0 0 0 0 2 4;3 0 0 1 5 0 0;4 0 0 1 2 2 4;
     5 0 0 1 6 2 3;6 0 0 1 3 0 0;1 2 1 0 0 0 0];
>> comp=scc(G)

comp =

    -1     2     3     2     3     3

>> [CFC,Ak]=cfc(comp,G);

Le componenti fortemente connesse con più di un nodo sono le
seguenti:

COMPONENTE =
     2     4

COMPONENTE =
     3     5     6

>> [G,loop]=elimina(G);
>> [CICLI]=algoritmo_Johnson(G,Ak,loop);
>> CICLI

CICLI =
    [1x3 double]
    [1x4 double]
    [1x3 double]
    [1x2 double]

>> CICLI{1,1}
ans =
     2     4     2

>> CICLI{2,1}
ans =
     3     5     6     3

>> CICLI{3,1}
ans =
     3     5     3

>> CICLI{4,1}
ans =
     4     4
```

Figura 5.8: Command Window di Matlab per la funzione algoritmo_Johnson applicato al grafo di Fig.5.7

Capitolo 6

Esempi di funzionamento del Toolbox e confronto tra i due metodi

In questo capitolo vengono presentati alcuni esempi per chiarire il funzionamento del toolbox Matlab implementato, viene inoltre preso in considerazione un modello fisico parametrico e ne viene studiata la diagnosticabilità sia con il software precedentemente implementato [22] sia con il software sviluppato in questa tesi.

6.1 Esempi di funzionamento del Toolbox

In questa sezione vengono presentati alcuni esempi di funzionamento del Toolbox per la diagnosticabilità sviluppato in questa tesi.

6.1.1 Esempio 1

Consideriamo la rete di Petri in Figura 6.1, in cui l'insieme delle transizioni osservabili è $T_o = \{t_1, t_2, t_3\}$, l'insieme delle transizioni non osservabili è $T_u =$

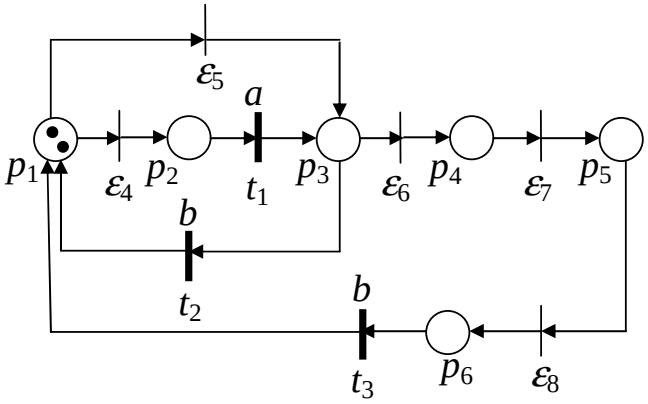


Figura 6.1: Esempio di rete di Petri con un'unica classe di guasto

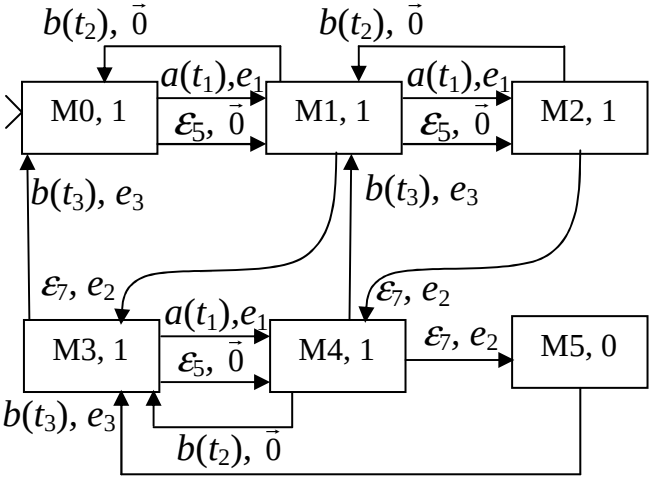


Figura 6.2: MBRG della rete di Petri in Figura 6.1

$\{\varepsilon_4, \varepsilon_5, \varepsilon_6, \varepsilon_7, \varepsilon_8\}$. L'unica classe di guasto è $T_f = \{\varepsilon_5, \varepsilon_7\}$. La funzione di etichettatura è tale che: $\mathscr{L}(t_1) = a$, $\mathscr{L}(t_2) = \mathscr{L}(t_3) = b$.

Il MBRG ottenuto tramite la funzione MBRG.m è mostrato in Figura 6.2. Il MBRD viene costruito richiamando la funzione:

[D,numTclas,sizeE,no,nf]= MBRD(T,Pre,Post,M0,F,L,E)

ed è mostrato in Figura 6.3. Osservando il MBRD è possibile individuare due cicli incerti per l'unica classe di guasto esistente (ellissi tratteggiate in rosso), ossia $\gamma = 5\,\underrightarrow{a}\,7\,\underrightarrow{b}\,5$ e $\gamma = 7\,\underrightarrow{a}\,9\,\underrightarrow{b}\,7$. A questo punto per verificare la diagnosticabilità del sistema è necessario andare ad individuare se tali cicli sono indeterminati oppure no. Bisogna prendere in considerazione le marcature present in ciascun nodo

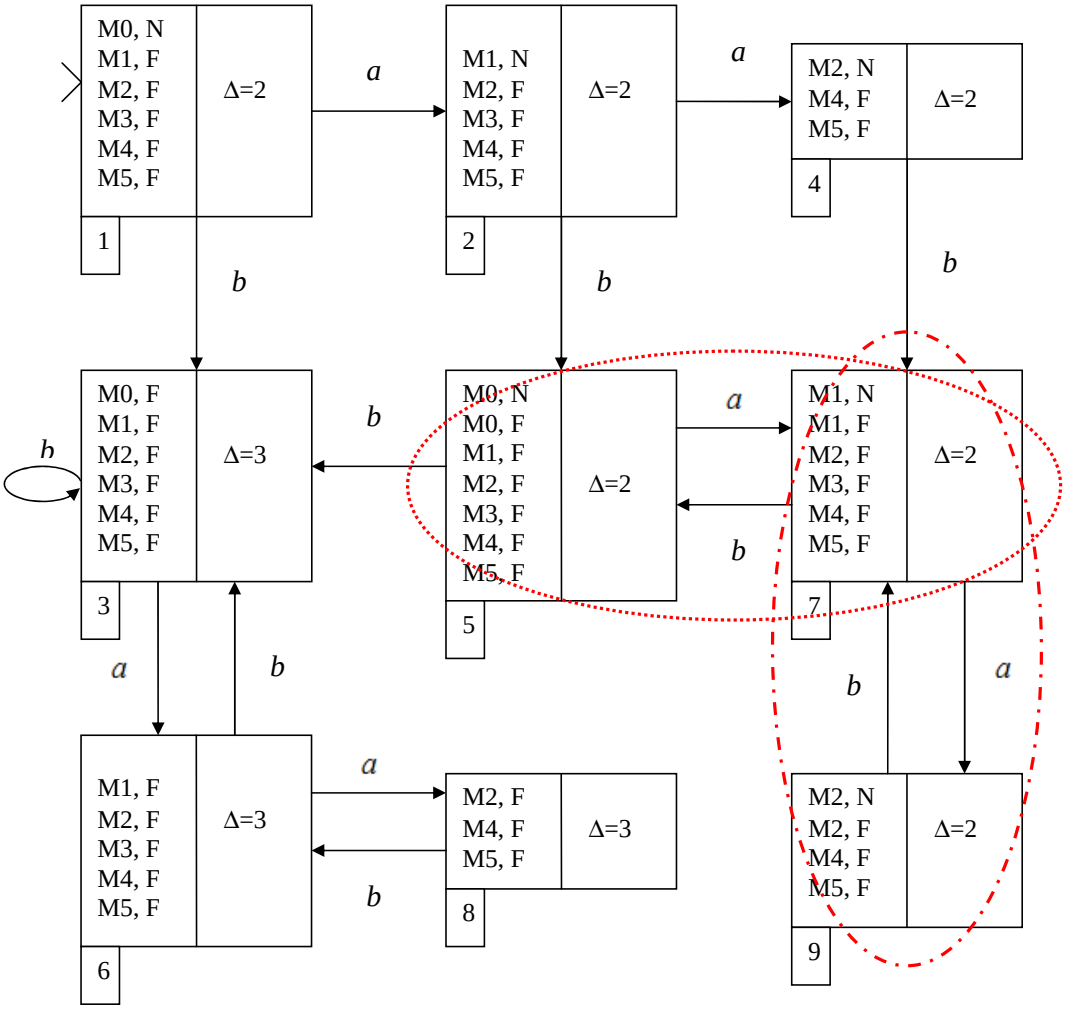


Figura 6.3: MBRD della rete di Petri in Figura 6.1

del ciclo ed andare ad individuare tutti i possibili cicli composti da marcature del MBRG con valore del vettore h pari a N, e tutti i possibili cicli composti da marcature del MBRG con valore del vettore h pari a F. Un ciclo è indeterminato se nel MBRG è presente almeno un ciclo N e almeno un ciclo F tra quelli possibili. Se consideriamo il ciclo incerto $\gamma = 5\,\underrightarrow{a}\,7\,\underrightarrow{b}\,5$, nel MBRG è possibili individuare il ciclo N $M_0\,\underrightarrow{a}\,M_1\,\underrightarrow{b}\,M_0$, ed anche un ciclo F $M_0\,\underrightarrow{a}\,M_1\,\underrightarrow{b}\,M_0$, di conseguenza tale ciclo è anche indeterminato.

Analizzando il ciclo incerto $\gamma = 7\,\underrightarrow{a}\,9\,\underrightarrow{b}\,7$, nel MBRG è possibili individuare il ciclo N $M_1\,\underrightarrow{a}\,M_2\,\underrightarrow{b}\,M_1$, ed anche un ciclo F $M_1\,\underrightarrow{a}\,M_2\,\underrightarrow{b}\,M_2$, di conseguenza anche questo ciclo è indeterminato.

```
>> MBRD_diag_ott(D,T,sizeE,no,nf)

********** CLASSE DI GUASTO 1 **********

Il sistema è non diagnosticabile in riferimento alla classe di
guasto 1 per la presenza di cicli indeterminati

*************************************

Il sistema non è diagnosticabile

*************************************
Elapsed time is 0.055112 seconds.
```

Figura 6.4: Command Windows di Matlab per la verifica della diagnosticabilità dell rete in Figura 6.1

Andiamo a verificare la diagnosticabilità tramite la funzione:

MBRD_diag_ott(D,MBRG,sizeE,no,nf)

il risultato fornito da tale funzione è mostrato in Figura 6.4. Come ci si aspettava la sola classe di guasto presente è non diagnosticabile, per cui il sistema è non diagnosticabile.

6.1.2 Esempio 2

Consideriamo la rete in Figura 6.5 in cui l'insieme delle transizioni osservabili è $T_o = \{t_1, t_2, t_3, t_4\}$, l'insieme delle transizioni non osservabili è $T_u = \{\varepsilon_5, \varepsilon_6\}$. Sono presenti due classi di guasto $T_f = T_f^1 \cup T_f^2 = \{\varepsilon_5\} \cup \{\varepsilon_6\}$. La funzione di etichettatura è tale che: $\mathscr{L}(t_1) = a$, $\mathscr{L}(t_2) = b$, $\mathscr{L}(t_3) = c$ e $\mathscr{L}(t_4) = d$.

Il MBRG corrispondente alla rete di Figura 6.5 è mostrato in Figura 6.6. Il MBRD costruito tramite la funzione MBRD.m è mostrato in Figura 6.7.

Nel grafo in Figura 6.7 sono messi in evidenza i cicli incerti relativi alla classe di guasto 1 (ellissi tratteggiate in rosso) e i cicli incerti relativi alla classe di guasto 2 (ellissi tratteggiate in verde). Più precisamente i cicli incerti per la prima classe di guasto sono: $\gamma = 1\,\underrightarrow{a}\,2\,\underrightarrow{b}\,1$ e $\gamma = 6\,\underrightarrow{d}\,6$, mentre i cicli incerti per la seconda classe di guasto sono: $\gamma = 2\,\underrightarrow{a}\,4\,\underrightarrow{b}\,2$ e $\gamma = 5\,\underrightarrow{c}\,5$. E' necessario stabilire se tali cicli sono indeterminati oppure no.

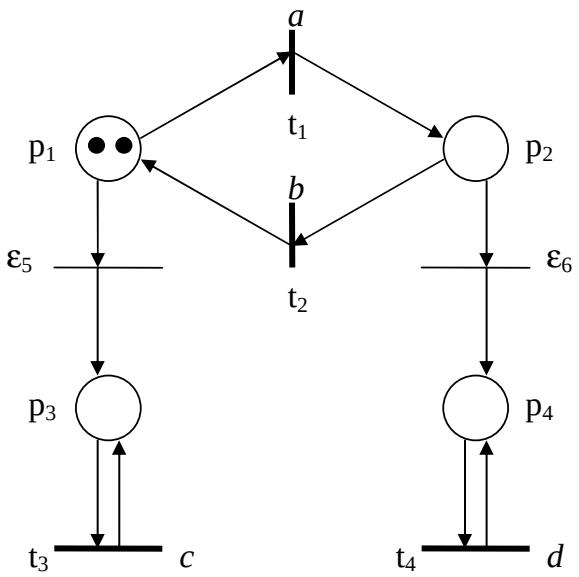


Figura 6.5: Esempio di rete di Petri con due classi di guasto

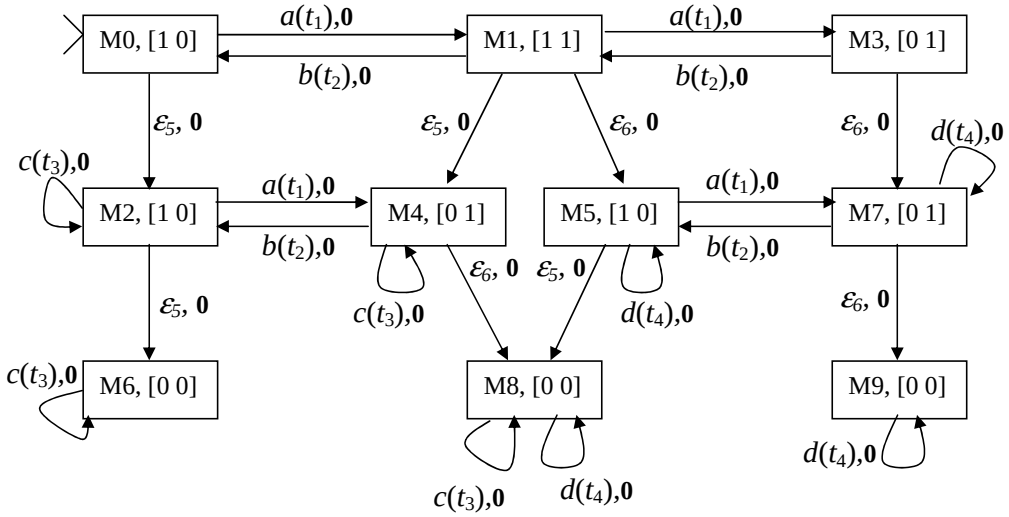


Figura 6.6: MBRG della rete di Petri in Figura 6.5

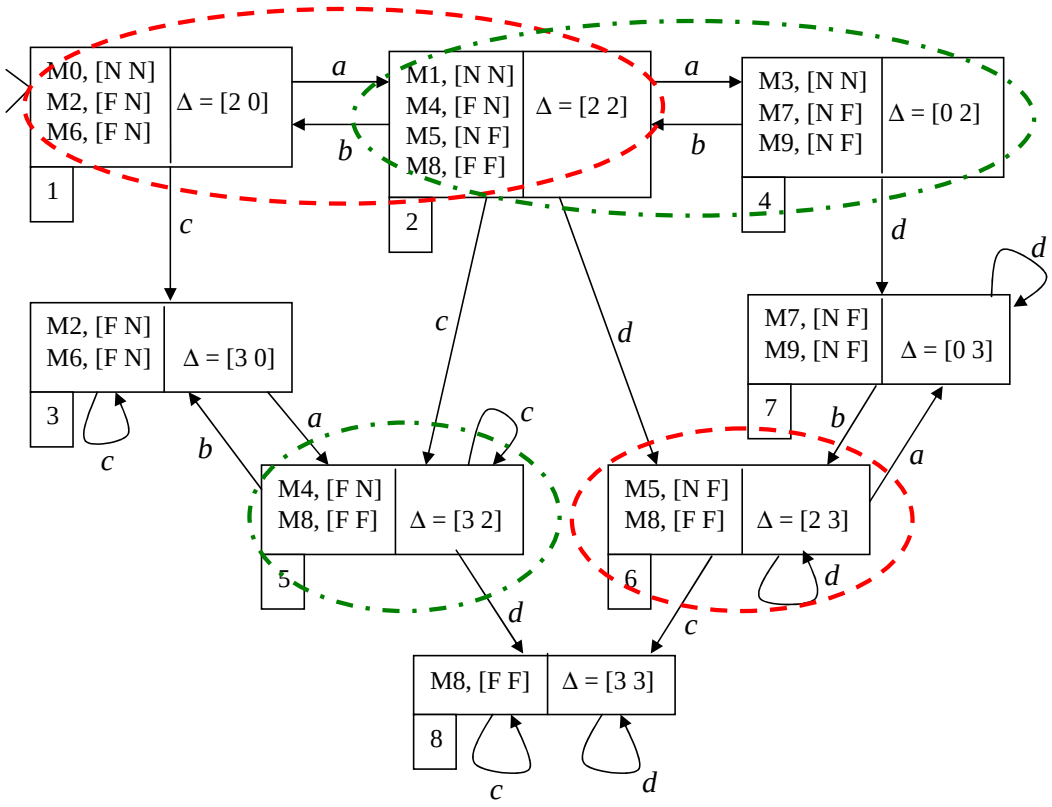


Figura 6.7: MBRD della rete di Petri in Figura 6.5

Analizziamo la prima classe di guasto. Consideriamo il ciclo incerto $\gamma = 1 \underrightarrow{a} 2 \underrightarrow{b} 1$, per stabilire se il ciclo è indeterminato è necessario verificare nel MBRG la presenza di almeno un ciclo N e almeno un ciclo F tra tutti i possibili cicli N e F ottenuti combinando le marcatura presenti nei nodi 1 e 2. Possiamo ad esempio individuare nel MBRG, il ciclo N, $M_0 \underrightarrow{a} M_1 \underrightarrow{b} M_0$ e il ciclo F, $M_2 \underrightarrow{a} M_4 \underrightarrow{b} M_2$. Possiamo dunque concludere che il ciclo incerto analizzato è indeterminato, di conseguenza la classe di guasto 1 è non diagnosticabile.

Analizziamo la seconda classe di guasto. Consideriamo il ciclo incerto $\gamma = 2 \underrightarrow{a} 4 \underrightarrow{b} 2$ e verifichiamo nel MBRG se esso è indeterminato. Possiamo ad esempio individuare nel MBRG, il ciclo N, $M_1 \underrightarrow{a} M_3 \underrightarrow{b} M_1$ e il ciclo F, $M_5 \underrightarrow{a} M_7 \underrightarrow{b} M_5$. Possiamo, anche in questo caso, concludere che il ciclo incerto analizzato è indeterminato, di conseguenza la classe di guasto 2 è non diagnosticabile.

In Figura 6.8 è mostrato il risultato ottenuto verificando la diagnosticabilità con la funzione MBRD_diag_ott.m.

```
>> MBRD_diag_ott(D,T,sizeE,no,nf)

********** CLASSE DI GUASTO 1 **********

Il sistema è non diagnosticabile in riferimento alla classe
di guasto 1 per la presenza di cicli indeterminati

********** CLASSE DI GUASTO 2 **********

Il sistema è non diagnosticabile in riferimento alla classe
di guasto 2 per la presenza di cicli indeterminati

************************************

Il sistema non è diagnosticabile

************************************
Elapsed time is 0.007917 seconds.
```

Figura 6.8: Command Windows di Matlab per la verifica della diagnosticabilità della rete in Figura 6.5

6.2 Test del software

Prendiamo in considerazione alcune reti di Petri e testiamo il software sviluppato in questa tesi, andando ad analizzare la cardinalità del MBRG e del MBRD, il loro tempo di calcolo e il tempo per il calcolo della diagnosticabilità, al variare del numero di gettoni, x nella marcatura iniziale.

Le tabelle con i risultati sono strutturate nel modo seguente:

- la colonna 1 indica il numero di gettoni della marcatura iniziale, x;
- le colonne 2 e 3 indicano rispettivamente la cardinalità del MBRG ($|MBRG|$) e il tempo di calcolo espresso in secondi (t_{MBRG});
- le colonne 4 e 5 indicano rispettivamente la cardinalità del MBRD ($|MBRD|$) e il tempo di calcolo espresso in secondi (t_{MBRD});

x	$\|MBRG\|$	$t_{MBRG}[s]$	$\|MBRD\|$	$t_{MBRD}[s]$	$t_{diag}[s]$	$Diag?$
1	4	0.01407	4	0.03955	0.010149	*Si*
2	10	0.03782	8	0.07709	0.15264	*No*
3	20	0.099	12	0.242	0.4413	*No*
4	35	0.1881	16	0.8104	1.0286	*No*
5	56	0.38052	20	2.845	3.3402	*No*
6	84	0.6639	24	11.309	12.0645	*No*

Tabella 6.1: Risultati delle simulazioni effettuate con il metodo proposto in questa tesi prendendo in considerazione la rete in Figura 6.5

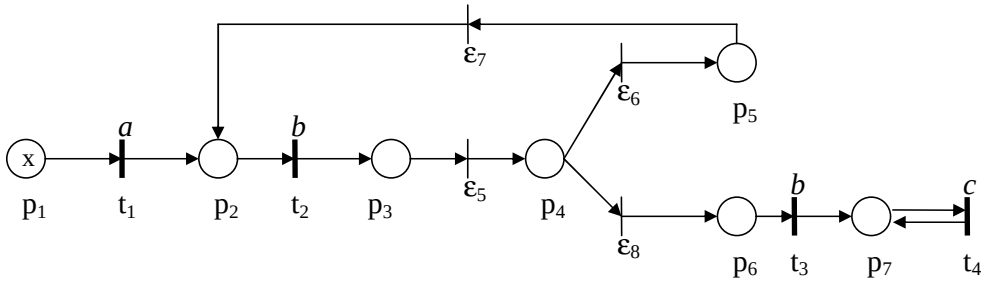


Figura 6.9: Esempio di rete di Petri per il test del software

- la colonna 6 indica il tempo di calcolo della diagnosticabilità espresso in secondi (t_{diag});
- la colonna 7 indica se il sistema è diagnosticabile oppure no.

Esempio 6.2.1. *Consideriamo nuovamente la rete in Figura 6.5, e facciamo variare il numero di gettoni x della marcatura iniziale da* 1 *a* 6*. I risultati ottenuti sono mostrati nella Tabella 6.1.* ■

Esempio 6.2.2. *Consideriamo la rete in Figura 6.9, in cui l'insieme delle transizioni osservabili è* $T_o = \{t_1, t_2, t_3, t_4\}$*,l'insieme delle transizioni non osservabili è* $T_u = \{\varepsilon_5, \varepsilon_6, \varepsilon_7, \varepsilon_8\}$*, le classi di guasto sono due ,ossia* $T_f^1 = \{\varepsilon_7\}$*,* $T_f^2 = \{\varepsilon_8\}$*. La funzione di etichettatura è:* $\mathscr{L}(t_1) = a, \mathscr{L}(t_2) = \mathscr{L}(t_3) = b$ *e* $\mathscr{L}(t_4) = c$*.*

Anche in questo caso facciamo variare il numero di gettoni x nella marcatura iniziale da 1 *a* 6*. I risultati ottenuti sono mostrati in Tabella 6.2.* ■

Esempio 6.2.3. *Consideriamo la rete in Figura 6.10, in cui l'insieme delle transizioni osservabili è* $T_o = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$*,l'insieme delle transizioni non osservabili è* $T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}\}$*, le classi di guasto sono due ,ossia* $T_f^1 =$

x	$\|MBRG\|$	$t_{MBRG}[s]$	$\|MBRD\|$	$t_{MBRD}[s]$	$t_{diag}[s]$	$Diag?$
1	5	0.01775	7	0.02588	0.04755	*Si*
2	15	0.06537	22	0.3355	0.4637	*No*
3	35	0.2168	50	2.908	3.1958	*No*
4	70	0.8308	95	22.818	23.977	*No*
5	126	2.3156	161	158.52	162.88	*No*
6	210	7.659	252	982.817	1002.54	*No*

Tabella 6.2: Risultati delle simulazioni effettuate con il metodo proposto in questa tesi prendendo in considerazione la rete in Figura 6.9

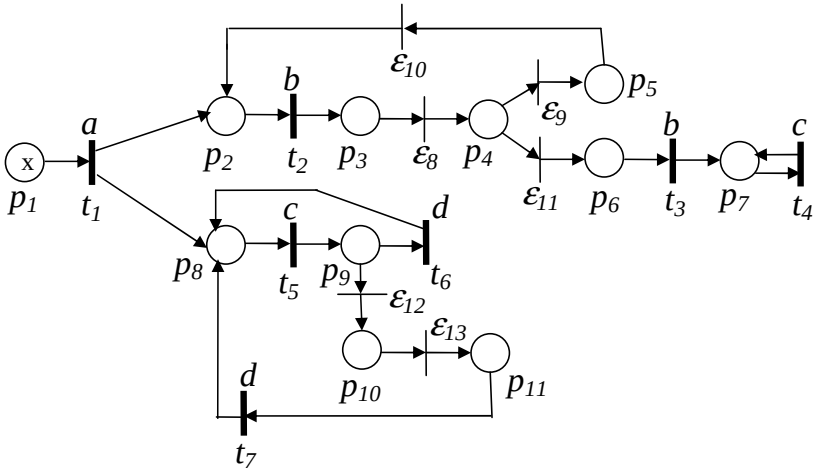


Figura 6.10: Esempio di rete di Petri per il test del software

$\{\varepsilon_{11}\}$, $T_f^2 = \{\varepsilon_{12}\}$. *La funzione di etichettatura è:* $\mathscr{L}(t_1) = a, \mathscr{L}(t_2) = \mathscr{L}(t_3) = b$, $\mathscr{L}(t_4) = \mathscr{L}(t_5) = c$ *e* $\mathscr{L}(t_6) = \mathscr{L}(t_7) = d$.

In questo caso facciamo variare il numero di gettoni x nella marcatura iniziale da 1 *a* 3. *I risultati ottenuti sono mostrati in Tabella 6.3.* ■

Osservando le Tabelle 6.1, 6.2 e 6.3 possiamo vedere come la cardinalità del MBRG e del MBRD aumenta ad ogni incremento del parametro x di una unità. Allo stesso modo crescono i tempi di calcolo sia dei due grafi sia il tempo di calcolo della diagnosticabilità, naturalmente man mano che la dimensione dei grafi aumenta i tempi di calcolo diventano sempre maggiori.

x	$\|MBRG\|$	$t_{MBRG}[s]$	$\|MBRD\|$	$t_{MBRD}[s]$	$t_{diag}[s]$	$Diag?$
1	13	0.092	20	0.3739	0.4958	No
2	73	1.983	142	50.463	53.743	No
3	273	43.659	750	6928.58	7894.2	No

Tabella 6.3: Risultati delle simulazioni effettuate con il metodo proposto in questa tesi alla rete in Figura 6.10

6.3 Confronto tra le due metodologie

6.3.1 Modello d'esempio

Il modello fisico considerato descrive una 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 poter 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. Affinchè 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.

Il sistema è caratterizzato da tre parametri:

- $2m$, il numero totale di linee di produzione;
- l, il numero di operazioni che devono essere effettuate su ciascun componente del prodotto;

- d, una variabile binaria che comporta alcune caratteristiche nella funzione di l'etichettatura. Nello specifico:
  $d = \begin{cases} 1, \text{ se il sistema è diagnosticabile} \\ 0, \text{ in caso contrario} \end{cases}$

Il modello analizzato è mostrato in Figura 6.11. Le transizioni osservabili sono evidenziate in verde, le transizioni non osservabili ma regolari sono evidenziate in blu, mentre quelle non osservabili di guasto in rosso. L'ellisse verde mette in evidenza la transizione osservabile a_{2-d}, dipendente dal parametro d. Se $d = 0$ la transizione sarà a_2, mentre se $d = 1$ la transizione sarà a_1.

6.3.2 Risultati numerici

Nella Tabella 6.4 sono mostrati i risultati ottenuti effettuando la simulazione con il software precedentemente implementato, in cui la diagnosticabilità veniva valutata tramite l'analisi di due grafi il MBRG e il BRD.

La tabella è strutturata in questo modo:

- le colonne 1, 2 e 3 indicano i valori assunti dai parametri m, l e d;
- la colonna 4 indica la cardinalità del grafo di raggiungibilità ($|R|$);
- le colonne 5 e 6 indicano rispettivamente la cardinalità del BRG ($|BRG|$) e il tempo impiegato per calcolarlo (t_{BRG}), espresso in secondi;
- le colonne 7 e 8 indicano rispettivamente la cardinalità del MBRG ($|MBRG|$) e il tempo impiegato per calcolarlo (t_{MBRG}), espresso in secondi;
- le colonne 9 e 10 indicano rispettivamente la cardinalità del BRD ($|BRD|$) e il tempo impiegato per calcolarlo (t_{BRD}), espresso in secondi;
- la colonna 11 indica il tempo impiegato per valutare la diagnosticabilità del sistema (t_{diag}), espresso in secondi.

Nella Tabella 6.5 sono mostrati i risultati ottenuti effettuando la simulazione con il software implementato in questa tesi, in cui la diagnosticabilità viene valutata tramite l'analisi di due grafi il MBRG e il MBRD.

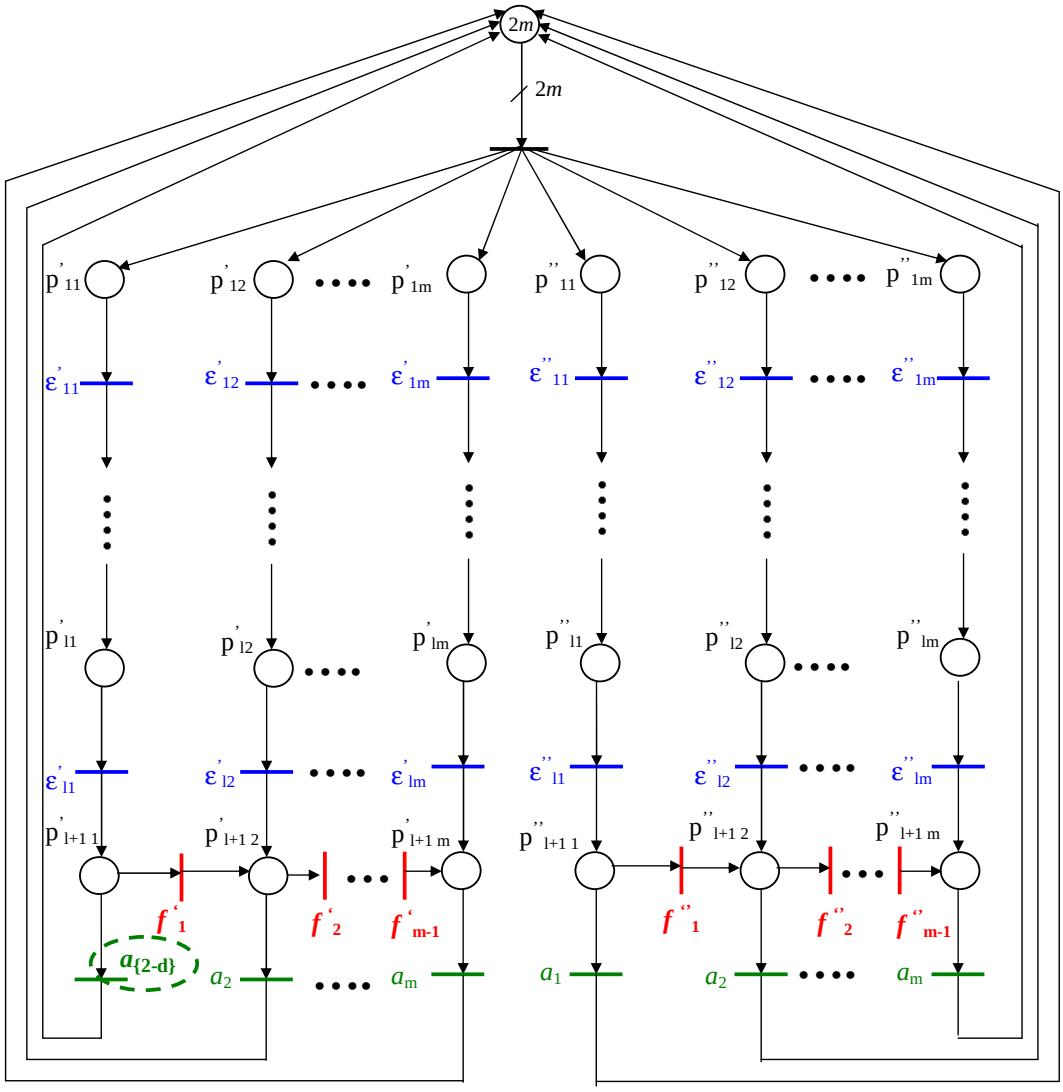


Figura 6.11: Modello parametrico

m	l	d	$\|R\|$	$\|BRG\|$	t_{BRG} [s]	$\|MBRG\|$	t_{MBRG} [s]	$\|BRD\|$	$t_{BRD}[s]$	$t_{diag}[s]$
2	1	0	121	16	0.2507	36	0.2489	23	0.3742	0.81924
2	1	1	121	16	0.1241	36	0.2286	19	0.1693	0.41527
2	2	0	361	16	0.8591	36	1.3919	23	0.9416	2.506
2	2	1	361	16	0.8951	36	1.4162	19	0.9416	2.4549
2	3	0	841	16	6.1768	36	8.9129	23	6.2582	15.2573
2	3	1	841	16	6.0672	36	8.8495	19	6.1128	14.9776
3	1	0	2025	64	21.997	484	54.796	67	24.0085	83.9516
3	1	1	2025	64	22.0254	484	55.225	55	23.266	3269.69
3	2	0	10000	64	1073.82	484	1952.07	67	1075.82	3032.92
3	2	1	10000	64	1063.88	484	1948.61	55	1065.13	6094.96
3	3	0	34225	64	25023.46	484	42769.45	67	25025.47	67800
3	3	1	34225	64	24037.15	484	41650.06	55	24038.54	68756.6

Tabella 6.4: Risultati delle simulazioni effettuate con il metodo proposto in [10]

m	l	d	$\|MBRG\|$	t_{MBRG} [s]	$\|MBRD\|$	$t_{MBRD}[s]$	$t_{diag}[s]$
2	1	0	36	0.2732	23	0.6637	10.9585
2	1	1	36	0.2324	18	0.3175	0.559042
2	2	0	36	1.3947	23	0.6232	11.8428
2	2	1	36	1.3975	18	0.3223	1.7317
2	3	0	36	9.0693	23	0.6096	19.4819
2	3	1	36	8.989	18	0.3157	9.30709
3	1	0	484	58.097	67	1148.85	520932.2
3	1	1	484	56.473	54	672.148	728.673
3	2	0	484	2009.37	67	1162.29	523640.04
3	2	1	484	2011.049	54	682.25	2694.006
3	3	0	484	41141.35	67	1168.19	563037.6
3	3	1	484	41188.35	54	684.63	41873

Tabella 6.5: Risultati delle simulazioni effettuate con il metodo proposto in questa tesi

La tabella è strutturata in questo modo:

- le colonne 1, 2 e 3 indicano i valori assunti dai parametri m, l e d;
- le colonne 4 e 5 indicano rispettivamente la cardinalità del MBRG ($|MBRG|$) e il tempo impiegato per calcolarlo (t_{MBRG}), espresso in secondi;

- le colonne 6 e 7 indicano rispettivamente la cardinalità del MBRD ($|MBRD|$) e il tempo impiegato per calcolarlo (t_{MBRD}), espresso in secondi;
- la colonna 8 indica il tempo impiegato per valutare la diagnosticabilità del sistema (t_{diag}), espresso in secondi.

Consideriamo le Tabella 6.4 e 6.5, è possibile fare le seguenti osservazioni:

- la cardinalità del BRG varia solo in funzione del parametro m mentre quella del grafo di raggiungibilità (R) varia sia in funzione di m che in funzione di l, inoltre si può osservare come la cardinalità del BRG sia sempre molto inferiore rispetto alla cardinalità di R;
- la cardinalità del BRG e del MBRG è invariante rispetto ai parametri l e d, mentre aumenta al crescere del parametro m, ossia del numero di linee di produzione di ciascun gruppo di lavoro;
- la cardinalità del BRD e del MBRD varia sia rispetto al parametro m sia rispetto al parametro d, ossia alla variabile che stabilisce se il sistema è diagnosticabile oppure no;
- per quanto riguarda i tempi di calcolo variano principalmente rispetto al numero di linee di produzione (m) e rispetto al numero di lavorazioni (l). Per $m = 2$ i tempi sono relativamente bassi, per $m = 3$ i tempi crescono notevolmente al crescere di l da 1 a 3.

Per quanto riguarda i tempi per il calcolo della diagnosticabilità, quello che si verifica è che per $d = 0$ (ossia quando il sistema è non diagnosticabile) si ottengono dei risultati migliori con il software precedentemente sviluppato mentre per $d = 1$ (ossia quando il sistema è diagnosticabile) si ottengono risultati migliori con il software sviluppato in questa tesi. Questo si verifica perché, nel caso in cui il sistema sia non diagnosticabile, con il software precedente viene analizzato un ciclo per volta e quindi è sufficiente che venga individuato un ciclo indeterminato nel MBRG per ogni classe di guasto per affermare che il sistema è non diagnosticabile. Nel nostro caso si ha una maggior perdita di tempo nella costruzione del MBRD (perchè bisogna considerare un numero maggiore di marcature) e nel calcolo dei cicli in quanto prima della verifica della diagnosticabilità per ogni classe di guasto, si ha la preventiva determinazione di tutti i cicli incerti nel MBRD per quella classe di guasto.

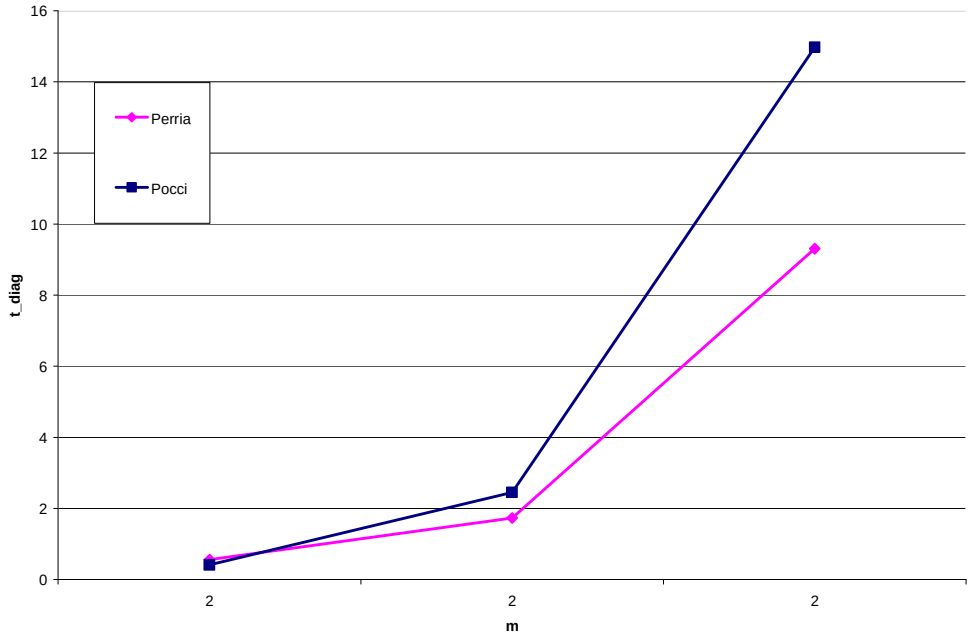


Figura 6.12: Confronto tra le due metodologie, considerando m=2, l che varia da 1 a 3 e d=1

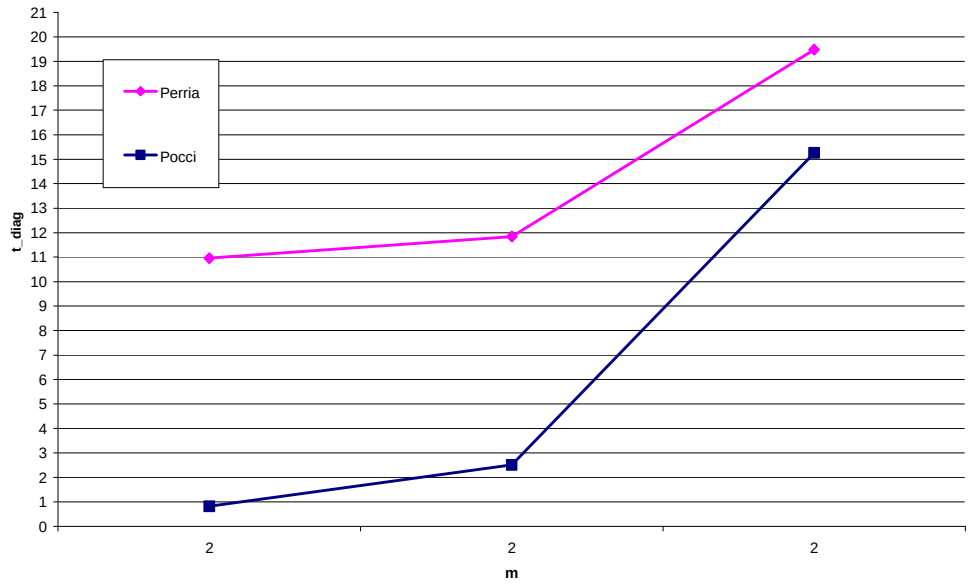


Figura 6.13: Confronto tra le due metodologie, considerando m=2, l che varia da 1 a 3 e d=0

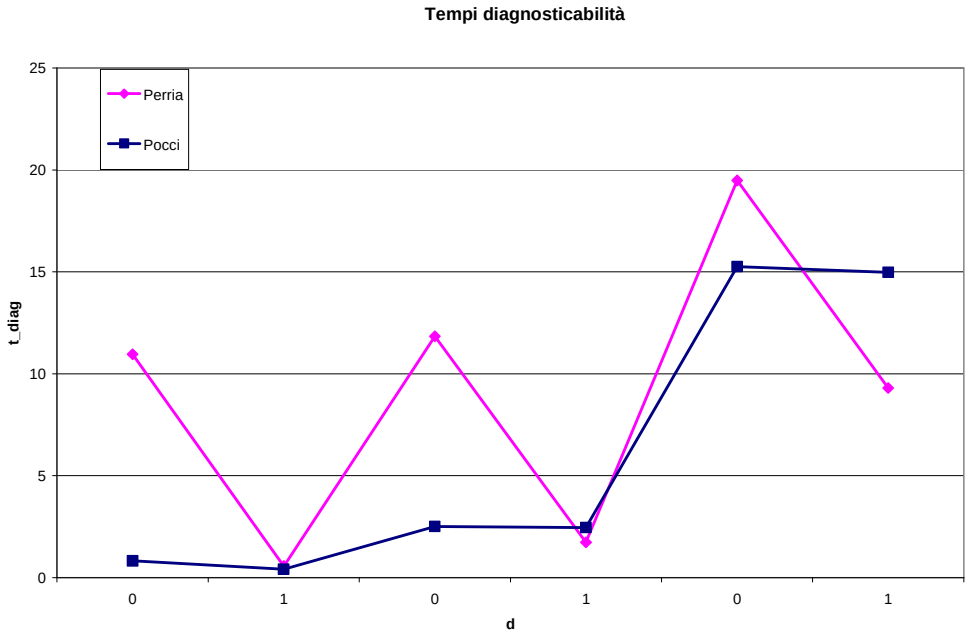


Figura 6.14: Confronto tra le due metodologie, considerando m=2, l che varia da 1 a 3 e d=0

Nel caso in cui il sistema sia diagnosticabile ($d = 1$) è necessario andare ad analizzare tutti i cicli presenti per poter affermare che lo sia, per cui il software precedente impiega un tempo maggiore in quanto è necessario andare a determinare anche i percorsi che conducono ai cicli.

Le considerazioni precedenti sono riscontrabili andando ad osservare i grafici in Figura 6.12, 6.13 e 6.14. In tutti i grafici la linea di colore blu indica i risultati ottenuti con il software sviluppato per la precedente metodologia ([22]), mentre la linea rosa indica i risultati ottenuti con il software sviluppato in questa tesi.

In Figura 6.12 è mostrato l'andamento del tempo di calcolo della diagnosticabilità per entrambe le metodologie, nel caso di $m = 2$, $l = (1,2,3)$ e per un valore di d pari a 1.

In Figura 6.13 è mostrato l'andamento del tempo di calcolo della diagnosticabilità per entrambe le metodologie, nel caso di $m = 2$, $l = (1,2,3)$ e per un valore di d, in questo caso, pari a 0.

In Figura 6.14 è mostrato l'andamento del tempo di calcolo della diagnosticabilità per entrambe le metodologie, nel caso di $m = 2$, $l = (1,2,3)$ e per d che varia da 0 a 1.

Capitolo 7

Conclusioni

In questa tesi abbiamo affrontato il problema della diagnosticabilità dei Sistemi ad Eventi Discreti, più precisamente sono state prese in considerazione del reti di Petri limitate.

Il lavoro da cui siamo partiti è quello di Cabasino *et al.* ([10]) relativo alla diagnosticabilità delle reti di Petri limitate. In tale lavoro viene presentato un approccio basato sul concetto di *marcatura di base*, e la verifica della diagnosticabilità è basata sull'analisi di due grafi il *Modified Basis Reachability Graph* e il *Basis Reachability Diagnoser*.

In questa tesi abbiamo proposto un metodo alternativo per la verifica della diagnosticabilità basato sull'analisi di due grafi il *Modified Basis Reachability Graph* e il *Modified Basis Reachability Diagnoser*. Questa nuova metodologia si proponeva come obiettivo un miglioramento dal punto di vista computazionale rispetto all'approccio precedente. Il nuovo metodo è stato inizialmente testato utilizzando la libreria UMDES per verificarne la correttezza.

E' stato scritto un programma per la costruzione del *Modified Basis Reachability Diagnoser* e un programma per la verifica della diagnosticabilità.

E' stato inoltre studiato ed implementato l'algoritmo di Johnson per il calcolo dei cicli elementari in un grafo orientato. Lo studio di tale algoritmo si è reso necessario per la verifica delle condizioni necessarie e sufficienti per la diagnosticabilità. E' stata fatta un'analisi della complessità computazionale dell'algoritmo di

Johnson.

Infine è stato testato il software implementato tramite alcuni esempi ed è stato eseguito un confronto tra l'approccio precedentemente trattato ([10]) e l'approccio proposto in questa tesi.

Appendice A

Listati dei programmi

In questo capitolo vengono inseriti i listati dei programmi principali che fanno parte del Toolbox di Matlab realizzato nel lavoro di tesi.

A.1 MBRD.m

```
% CALCOLO DEL MODIFIED BASIS REACHABILITY DIAGNOSER (MBRD) A PARTIRE DAL
% MODIFIED BASIS REACHABILITY GRAPH (MBRG)

function [D,numTclas,sizeE,no,nf] = MBRD(T,Pre,Post,M0,F,L,E)

% La funzione MBRD prende in ingresso la matrice T che corrisponde
% all'uscita della funzione MBRG.m, le matrici Pre e Post della rete in
% esame, la marcatura iniziale e tre vettori di celle F,L,E.
% Il vettore di celle F ha tante righe quanto è il numero classi di guasto
% e contiene in ogni riga le transizione appartenenti alla medesima classe
% di guasto. Il vettore di celle L ha un numero di righe pari alla
% cardinalità dell'alfabeto considerato e contiene in ciascuna riga le
% transizioni osservabili aventi la stessa etichetta. Nel vettore di celle E
% ogni riga corrisponde ad una differente etichetta nell'alfabeto considerato.
%La matrice di celle D in uscita ha la seguente struttura:
 %          ___________________________________________________________________
 %         |      1      |              2                      |    3          |
 %         |_____________|_____________________________________|_______________|
 %     D=  |    #nodo    | [#nodo MBRG,Marcatura,{N,F}]        |  etichetta/   |
 %         |             |                                     |nodo raggiunto|
 %         |_____________|_____________________________________|_______________|
 %           ____________________________
 %          |          4          |   5   |
 %          |_____________________|_______|
 %          |      Stato di       | Tag   |
 %          |      diagnosi       |       |
 %          |_____________________|_______|
```

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

%************************************************************************
%VERIFICA DEI PARAMETRI IN INGRESSO
%************************************************************************

if (size(Post)==size(Pre))&(c==1)&(p==m), elseif
size(Post)~=size(Pre)
   fprintf('\n ERRORE! Le matrici Pre e Post hanno dimensione differente\n')
else
   fprintf('\n  ERRORE! La dimensione della marcatura non è corretta\n')
end

if (lenghtL==1), else
   fprintf('\n  ERRORE nella dimensione della matrice L\n')
end

if (b==1), else
   fprintf('\n  ERRORE nella dimensione della matrice F\n')
end

if (size(L)==size(E)), else
   fprintf('\n  ERRORE Le matrici L ed E hanno dimensione differente\n')
end

%************************************************************************
% DETERMINAZIONE DEL 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});
%ogni riga di L può contenere
%più transizioni(aventi la stessa etichetta) per poter valutare il num
%di transizioni osservabili è necessario analizzare il vettore di celle L
%riga per riga
    no=no+lenghtLi;
end

nf=0;
for i=1:numTclas
%ogni riga di F può contenere
%più transizioni di guasto(facenti parte della medesima classe di guasto) per
%poter valutare il num. di transizioni di guasto è necessario analizzare il
%vettore di celle F riga per riga
    [sizeFi,lenghtFi]=size(F{i,1});
    nf=nf+lenghtFi;
end

nreg=n-no-nf;
%ovviamente il numero di transizioni regolari sarà dato dal num. totale di
%transizioni meno il num. di transizioni osservabili meno il num. di
```

```
%transizioni di guasto nu=nreg+nf;


%*************************************************************************
%INIZIALIZZAZIONE DEL MODIFIED BASIS REACHABILITY DIAGNOSER
%*************************************************************************

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

%Bisogna far attenzione al fatto che il primo nodo del MBRD comprenderà la
%marcatura iniziale M0 più tutte quelle marcature raggiunte a partire da M0
%con lo scatto di transizioni di guasto. Nel caso in cui da M0 non siano
%abilitate transizioni di guasto allora il primo nodo comprenderà solo la
%marcatura M0

%*************************************************************************
%INIZIO COSTRUZIONE DEL NODO INIZIALE
%*************************************************************************

if (T{1,4}(no+1:no+nf)==0)
%se dal nodo iniziale dell'MBRG non sono abilitate transizioni di guasto,
%il nodo iniziale dell'MBRD comprenderà solo la marcatura M0 e il vettore h
%assumerà per tutte le classi di guasto il valore N(-1) stando ad indicare
%che non sono scattate transizioni di guasto

    D{1,2}(1,1)=1; %numero del nodo dell'MBRG
    D{1,2}(1,2:p+1)=M0';
    for i=1:numTclas
        D{1,2}(1,p+i+1)=-1;
        D{1,4}(1,i)=0;
    end
else

%devo valutare se dalla marcatura iniziale sono abilitate transizioni
%di guasto e in tal caso a quale nodo (marcatura) mi conducono

   D{1,2}(1,1)=1;
   D{1,2}(1,2:p+1)=M0';
   for i=1:numTclas
        D{1,2}(1,p+1+i)=-1;
   end

   M=[zeros(1000,1+numTclas)];
   M(1,1)=1;
   M(1,2:numTclas)=0;
   countM=1;
   while(find(M(countM,1)~=0))
       Lf=0;
       Lff=0;
       for f=1:numTclas
       [sizeFi,lengthFi]=size(F{f,1});
       vv=find(T{M(countM,1),4}(1,no+1+Lf:no+Lf+lengthFi)==1);
       %vv è il vettore contenente gli indici delle transizioni di guasto
       %abilitate da nodo M(countM,1)
       if(vv==0),
       %significa che non ci sono transizioni di guasto abilitate e
       %non faccio nulla
       else
       [rvv,cvv]=size(vv);
```

```
       n=find(M(:,1)~=0);
       [sizen,lengthn]=size(n);
       for i=1:cvv
       nodo=T{M(countM,1),5}{1,no+Lff+vv(i)}{1,1}{1,1};
       %mi restituisce il numero del nodo raggiunto con lo scatto della
       %transizione di guasto no+Lff+vv(i)
       V=zeros(1,1+numTclas);
       V(1,1)=nodo;
       V(1,1+f)=1;

       for ff=1:numTclas
       if (M(countM,1+ff)==0),
       else
           V(1,1+ff)=1;
       end
       end

       if (V(1,:)-M(countM,:)==0),
       else
       M(sizen+i,1)=nodo;
       M(sizen+i,1+f)=1;

       for ff=1:numTclas
       if (M(countM,1+ff)==0),
       else
           M(sizen+i,1+ff)=1;
       end
       end
       end

       end
      end
       Lff=Lf+lengthFi;
       Lf=lengthFi;

   end


 %inserisco nel nodo iniziale le marcature raggiunte con lo scatto di
 %transizioni di guasto, che ricavo dalla matrice M

     [sizeD,lengthD]=size(D{1,2});
     [sizeM,lengthM]=size(M);
     for j=1:sizeM
         if (M(j,1)==0),
         else
         D{1,2}(sizeD+j,1)=M(j,1);
         D{1,2}(sizeD+j,2:p+1)=T{M(j,1),2};
         for jj=1:numTclas
            if(M(j,1+jj)==0)
            D{1,2}(sizeD+j,p+1+jj)=-1;
            else
            D{1,2}(sizeD+j,p+1+jj)=-2;
            end
         end
         end
     end
    countM=countM+1;
end
```

```
%Elimino le marcature doppie eventualmente presenti nel nodo iniziale

    [sizeD,lengthD]=size(D{1,2});
        for i=1:sizeD
            for j=1:sizeD
                if (i~=j &(D{1,2}(i,:)==D{1,2}(j,:)))
                    D{1,2}(j,:)=-3;
    %modifico tutte le colonne della riga j di Dnew{1,2} assegnandogli un
    %valore di riconoscimentoquesto mi indica che quella riga è già presente
                end
            end
        end
        A=[];
        countM=0;
        for j=1:sizeD
            if (D{1,2}(j,:)~=-3)
                countM=countM+1;
                A(countM,:)=D{1,2}(j,:);
            end
        end

    D{1,2}=A; %ho eliminato le eventuali righe ridondanti

%STATO DI DIAGNOSI DEL NODO INIZIALE

   H=D{1,2}(:,p+2:p+1+numTclas);
   for f=1:numTclas
           if (H(:,f)==-1)
            D{1,4}(1,f)=0;
           elseif(H(:,f)==-2)
                D{1,4}(1,f)=3;
           else
                D{1,4}(1,f)=2;

           end
   end

end

%*************************************************************************
%FINE COSTRUZIONE DEL NODO INIZIALE
%*************************************************************************

%*************************************************************************
%COSTRUZIONE DEGLI ALTRI NODI DELL'MBRD
%*************************************************************************

c=cell2mat(D(:,5)); d=find(c==0); while ~isempty(d)
    d=find(c==0);
    for z=1:sizeL %per ogni etichetta
        clear Dnew
        count=0;
        [sizeDi,lengthDi]=size(D{d(1,1),2});
        %dimensioni del nodo senza tag preso in considerazione.
        %sizeDi mi dice il numero di marcature presenti in questo nodo
        Dnew=[{[]}, {[]}, {[]}, {[]}, {0}];
        %matrice di celle temporanea in cui memorizzare le
        %nuove marcature raggiunte
```

```
    Dnew{1,2}=zeros(1,1+p+numTclas);
    Dnew{1,3}=empty_vector(sizeL);

    for m=1:sizeDi    %per ogni marcatura presente in D{d(1,1),2}
        nodocorr=D{d(1,1),2}(m,1);
        %numero del nodo dell'MBRG corrispondente alla marcatura m
        [numofarcmax,numoflabel]=size(T{nodocorr,8});
        Hcorrente=D{d(1,1),2}(m,p+2:p+1+numTclas);
        %vettore h associato alla marcatura m
        if(T{nodocorr,7}(1,z)==1)
          %se l'etichetta z-esima è abilitata
           for i=1:numofarcmax
               if (size(T{nodocorr,8}{i,z})~=[0,0])
                   count=count+1;
                   nodoragg=T{nodocorr,8}{i,z}{1,3}{1,1};
               %nodo raggiunto con lo scatto dell'etichetta z
                   Mragg=(T{nodoragg,2})';
               %marcatura corrispondente al nodo raggiunto

                   Dnew{1,2}(count,1:p+1)=[nodoragg Mragg];
                   %aggiorno Dnew{1,2}
                   for f=1:numTclas
                   if (Hcorrente(f)==-1)
        % il valore di h della nuova marcatura raggiunta resta invariato
        %in quanto non si è verificato guasto
                    Dnew{1,2}(count,p+1+f)=-1;
                     else
                    Dnew{1,2}(count,p+1+f)=-2;
                   end
                   end
                else
                    continue
               end
            end
     % Se dalla marcatura corrente non è abilitata l'etichetta z-esima,
     % passo alla marcatura successiva
        else
          continue
        end
    end

    if(Dnew{1,2}==0)
    %se il valore di Dnew{1,2} rimane impostata al valore di partenza
        continue
    %prosegui analizzando un nodo non ancora esplorato
    end

    %a questo nuovo nodo devo aggiungere le marcature raggiunte
    %con lo scatto di transizioni di guasto

        [sizeDn,lengthDn]=size(Dnew{1,2});
    %sideDn mi da il numero di marcature raggiunte con lo scatto
    %dell'etichetta z

        for s=1:sizeDn
       %per ogni marcatura raggiunta con lo scatto della etichetta z

        M=[zeros(1000,1+numTclas)];
%creo una matrice di zeri, avente un numero di colonne pari al
```

```
%numero di classi di guasto più una.L'elemento in prima clnna
%indic il numero del nodo dell'MBRG, gli elementi nelle
%restanti colonne valgono uno o zero a seconda che la
%transizione di guasto si verifichi oppure no
        nragg=Dnew{1,2}(s,1:1);
        % numero del nodo dell'MBRG corrispondente
        M(1,1)=nragg;
        for f=1:numTclas
            if (Dnew{1,2}(s,p+1+f)==-1)
%se vale -1 significa che la classe di guasto f non è scattata
                M(1,1+f)=0;
%quindi metto a 0 l'elemento relativo a nragg nella riga f+1
            else
                M(1,1+f)=1;
            end
        end
        countM=0;
        while(find(M(countM+1,1)~=0))
        countM=countM+1;
        Lf=0;
        Lff=0;
        for f=1:numTclas
        [sizeFi,lengthFi]=size(F{f,1});
        vv=find(T{M(countM,1),4}(1,no+1+Lf:no+Lf+lengthFi)==1);
        %vettore contenente gli indici delle transizioni di guasto
        %abilitate da quel nodo
        if(vv==0),
        else
        [rvv,cvv]=size(vv);
        n=find(M(:,1)~=0);
        [sizen,lengthn]=size(n);
        for i=1:cvv
        nodo=T{M(countM,1),5}{1,no+Lff+vv(i)}{1,1}{1,1};
        %mi restituisce il numero del nodo raggiunto con lo
        %scatto della transizione di guasto no+Lff+vv(i)

        V=zeros(1,1+numTclas);
        %vettore utile per evitare di finire in un ciclo infinito
        %nel caso di presenza di loop di transizioni di guasto
        V(1,1)=nodo;
        V(1,1+f)=1;

        for ff=1:numTclas
          if (M(countM,1+ff)==0),
          else
          V(1,1+ff)=1;
          end
        end
        if (V(1,:)-M(countM,:)==0),
        else
        M(sizen+i,1)=nodo;
        M(sizen+i,1+f)=1;
        for ff=1:numTclas
        if (M(countM,1+ff)==0),
        else
        M(sizen+i,1+ff)=1;
        end
        end
        end
```

```
            end

            end
      %M
            Lff=Lf+lengthFi;
            Lf=lengthFi;
       end
   end

     [sizeD,lengthD]=size(Dnew{1,2});
     [sizeM,lengthM]=size(M);
     for j=1:sizeM
         if (M(j,1)==0),
         else
        Dnew{1,2}(sizeD+j,1)=M(j,1);
        Dnew{1,2}(sizeD+j,2:p+1)=T{M(j,1),2};
        for jj=1:numTclas
          if(M(j,1+jj)==0)
          Dnew{1,2}(sizeD+j,p+1+jj)=-1;
          else
          Dnew{1,2}(sizeD+j,p+1+jj)=-2;
          end
        end
        end
     end
end

      %nel nodo creato posso essere presenti marcature identiche
      %caratterizzate dallo stesso valore di h, per cui è necessario
      %eliminare le ridondanze


        [sizeDnew,lengthDnew]=size(Dnew{1,2});
        for i=1:sizeDnew
            for j=1:sizeDnew
                if (i~=j &(Dnew{1,2}(i,:)==Dnew{1,2}(j,:)))
                    Dnew{1,2}(j,:)=-3;
    %modifico tutte le colonne della riga j di Dnew{1,2} assegnandogli un
    %valore di riconoscimentoquesto mi indica che quella riga è già presente
                end
            end
        end
        A=[];
        countM=0;
        for j=1:sizeDnew
            if (Dnew{1,2}(j,:)~=-3)
                countM=countM+1;
                A(countM,:)=Dnew{1,2}(j,:);
            end
        end

        Dnew{1,2}=A; %ho eliminato le eventuali righe ridondanti

        %Definizione dello stato di diagnosi per ogni classe di guasto
        %delta=0 se tutte le marcature presenti nel nodo hanno h=N (h=-1)
        %delta=2 se ci sono sia marcature con h=N che con h=F
        %delta=3 se tutte le marcature hanno h=F (h=-2)

       [sizeDnew, lengthDnew]=size(Dnew{1,2});
```

```
       H=Dnew{1,2}(:,p+2:p+1+numTclas);
       for f=1:numTclas
            if (H(:,f)==-1)
                Dnew{1,4}{1}(1,f)=0;
            elseif(H(:,f)==-2)
                Dnew{1,4}{1}(1,f)=3;
            else
                Dnew{1,4}{1}(1,f)=2;
            end
       end

%Bisogna ora verificare se il nodo appena definito è già presente
%nel grafo oppure no

        DDnew=Dnew{1,2};
        DDold=[];
        % Determino il numero di nodi del BRD (numofnodeD)
        [numofnodeD,lengthD]=size(D);
        [sizeDDnew,lenghtDDnew]=size(DDnew);
        exist=0;
        for i=1:numofnodeD
            DDold=D{i,2};
            [sizeDDold,lenghtDDold]=size(DDold);

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

            if (sizeDDold==sizeDDnew)
                for j=1:sizeDDnew
                    for jj=1:sizeDDold
                        if (DDnew(j,:)==DDold(jj,:))
                            DDold(jj,:)=-10;
                        %break
                        end
                    end
                end


                if(DDold==-10)
                    D{d(1,1),3}{z}{2}=i;
                    D{d(1,1),3}{z}{1}=E{z,1};
                    exist=1;
                    break;
                end
            end
        end

        if(exist==1) %il nodo esiste già
            continue
        else  %aggiungi il nuovo nodo
            [numofnodeD,lengthD]=size(D);
            D{numofnodeD+1,1}=numofnodeD+1;
            D{numofnodeD+1,2}=Dnew{1,2};
            D{d(1,1),3}{z}{1}=E{z,1};
            D{d(1,1),3}{z}{2}=numofnodeD+1;
            D{numofnodeD+1,4}=Dnew{1,4}{1};
            D{numofnodeD+1,5}=0;
        end
    end
```

```
    %per in nodo appena esplorato modifico il valore del tag da 0 a 1 e
    %continuo con gli eveentuali nodi non esplorati
    D{d(1,1),5}=1;
    c=cell2mat(D(:,5));
    d=find(c==0);
end end
```

A.2 showMBRD.m

```
%************************************************************************

%Funzione showMBRD che permette di ricavare facilmente la struttura del
%MBRD estraendo tutte le informazioni dalla matrice di celle restituita
%dalla funzione MBRD.m

%************************************************************************



function showMBRD(D,numTclas)

%Numero di nodi dell'MBRD
[sizeD,lengthD]=size(D); n=sizeD;

%Visualizzazione del numero di nodi
fprintf('\nThe number of nodes in the Modified Basis Reachability
Diagnoser is:
        %d\n\n',n)

for nodo=1:n %ciclo for sul numero totale di nodi
%Nodo corrente
    fprintf('\n # Node N%d',nodo)


%stato di diagnosi
    fprintf('\tDelta = [')
    for f=1:numTclas
       fprintf('%d ',D{nodo,4}(1,f))
    end
    fprintf('\b]\t\b')

%Numero di marcature appartenenti al nodo corrente
[numM,lengthM]=size(D{nodo,2});

fprintf('\n\t\b The markings belonging to the node and the
corresponding vectors h are:\t\b')

for m=1:numM  %for sul numero di marcature presenti nel nodo
indice_marc=D{nodo,2}(m,1);
fprintf('\n\t\b Marking M%d = [',indice_marc-1)
```

```
%Mi serve ricavare il numero di posti della rete
[sizeDi,lengthDi]=size(D{1,2}); p=lengthDi-1-numTclas;
%perchè ogni riga della seconda cella di D ha dimensione (1+p+numTclas)

for j=2:p+1
    fprintf('%d ',D{nodo,2}(m,j))
end fprintf('\b]\t\b')

%Visualizzazione del vettore h

fprintf('    h = [') for jj=p+2:p+1+numTclas
    if(D{nodo,2}(m,jj)==-1)
        fprintf('N ')
    elseif (D{nodo,2}(m,jj)==-2)
        fprintf('F ')
    end
end fprintf('\b]') end

%Visualizzazione delle etichette abilitate dal nodo considerato e del
%relativo nodo raggiunto

nessun_arco=1;

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

[a,b]=size(D{nodo,3});
% b è il numero di transizioni abilitate dal nodo

for t=1:b
    [c,d]=size(D{nodo,3}{t});
    if c~=0
        nessun_arco=0;
        etichetta=D{nodo,3}{1,t}{1,1};
        nodo_raggiunto=D{nodo,3}{1,t}{1,2};
        fprintf(' %s ---> N%d\n',etichetta,nodo_raggiunto)
    end
end

if nessun_arco==1
    fprintf('\t\b There are no labels enabled to fire\n\t\b')
end

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

A.3 algoritmo_Johnson.m

```
%************************************************************************

%Questa funzione calcola i cicli all'interno di un grafo orientato. Il
%calcolo dei cicli è basato sull'Algoritmo do Johnson (vedi articolo:
%FINDING ALL THE ELEMENTARY CIRCUITS OF A DIRECTED GRAPH - D. B. JOHNSON)
%Tale funzione richiama al suo interno la funzione CIRCUIT, che a sua volta
```

```
%richiama al suo interno se stessa ricorsivamente e la funzione UNBLOCK.
%Anche la funzione UNBLOCK richiama ricorsivamente se stessa.

%************************************************************************


function [CICLI]=algoritmo_Johnson(G,Ak,loop)

CICLI=[]; [n,lengthG]=size(G);
%n-1 è il numero dei nodi

for i=1:n-1 B{i,1}=i; end

blocked=zeros(1,n-1);
%i valori del vettore blocked vengono inizialmente settati
%al valore 0, ossia "false"

[sizeAk,lengthAk]=size(Ak);
%sizeAk mi da il numero di componenti fortemente connesse

for k=1:sizeAk
%per ogni componente fortemente connessa
    stack=[];
    s=1;
    A=Ak{k,1};

while (s<n-1)
        Vk=[];
        [sizeA,lengthA]=size(A);
        if(sizeA~=1)
            for j=1:sizeA
            Vk(j)=A(j,1);
%il vettore Vk contiene il valore dei nodi della componente fortemente
%connessa in esame
            end
            s=min(Vk);
%prendo come non s il più piccolo dei nodi di Vk
            for jj=1:length(Vk)
                blocked(Vk(jj))=0;
                B{Vk(jj),2}=[];
            end
            [circuit,CICLI]=CIRCUIT(blocked,stack,A,s,s,CICLI,B);

%elimino dalla componente fortemente connessa in analisi
%il nodo s e gli archi entranti e uscenti da questo nodo

            A(1,:)=0;
            for j=1:sizeA
            ind=(find(A(j,2:lengthA)==s));
            if isempty(ind),
            else
            [sizeind,lengthind]=size(ind);
            for jj=1:lengthind
            A(j,1+ind(jj))=0;
            end
            end
            end
            s=s+1;
        else
```

```
            s=n-1;
        end
        A=A(2:sizeA,:);
end end

[sizeC,lengthC]=size(CICLI); [sizel,lengthl]=size(loop);

%aggiungo gli eventuali loop individuati tramite la funzione elimina
%ai cicli trovati

for j=1:sizel
 CICLI{sizeC+j,1}=loop(j,:);
end

%la matrice di celle CICLI potrebbe contenere delle celle vuote,
%se così fosse le elimino mantenendo solo le celle contenenti i cicli

[sizeC,lengthC]=size(CICLI);

elimina=zeros(1,sizeC); for j=1:sizeC
    if(size(CICLI{j,1})==[0,0])
    elimina(1,j)=1;
    end
end

C=[]; conta=1; for j=1:sizeC
    if elimina(1,j)==1,
    else
        C{conta,1}=CICLI{j,1};
        conta=conta+1;
    end
end CICLI=C;

end
```

A.3.1 UNBLOCK.m

```
function [blocked]=UNBLOCK(blocked,u,B)

blocked(1,u)=0;  %0=false 1=true
[sizeB,lengthB]=size(B{u,2});

for i=1:lengthB
    ww=B{u,2}(1,i);
        if(blocked(ww)==0),
        else
            [blocked]=UNBLOCK(blocked,ww,B);
        end
end

B{u,2}=[];

end
```

A.3.2 CIRCUIT.m

```
function [circuit,CICLI]=CIRCUIT(blocked,stack,A,s,v,CICLI,B)

[sizeC,lengthC]=size(CICLI); f=0;
stack=[stack v]; %aggiungo il nodo v alla pila

blocked(v)=1;

%blocco il nodo v, nel momento in cui il nodo v viene inserito nella
%pila e bloccato, può essere inserito nuovamente nella pila solo se
%viene sbloccato

[sizeA,lengthA]=size(A); count=1;

for i=1:lengthA-1
%devo prendere in considerazione ogni nodo w appartenente alla
%lista di adiacenza del nodo v
    d=find(A(:,1)==v);
    w=A(d,1+i);
    if(w==0),
    elseif(w==s)

%quando w = s, ossia coincide con il nodo iniziale della pila,
%ho individuato un ciclo

    CICLI{sizeC+count,1}=[stack s];
    CICLI{sizeC+count,1};
    if(i==lengthA-1)
        f=1;
    end
    elseif (blocked(w)==0)
        [circuit,CICLI]=CIRCUIT(blocked,stack,A,s,w,CICLI,B);
        if(circuit==1)
            f=1;
        end

    if(f==1)
        [blocked]=UNBLOCK(blocked,v,B);
       for l=1:length(stack)
           blocked(stack(l))=1;
       end
    else

        for j=1:lengthA-1
        d=find(A(:,1)==v);
        w=A(d,1+j);
        if(w==0),
        else
        B{w,2};
        dd=find(B{w,2}==v);
        if(isempty(dd))
            [sizeB,lengthB]=size(B{w,2});
            B{w,2}(1,lengthB+1)=v;
        end
        end
        end
    end
```

```
    end
    count=count+1;
 end

    L=length(stack);
    stack=stack(1:L-1);
    circuit=f;

end
```

A.4 MBRD_diag_ott.m

```
%Funzione che stabilisce se il sistema e diagnosticabile o meno.
%La diagnosticabilità è basata sulla determinazione dei cicli incerti
%nell'MBRD e la verifica nell'MBRG del fatto che questi cicli siano o no
%indeterminati.

function MBRD_diag_ott(D,MBRG,sizeE,no,nf)

%la dignosticabilità va valutata per ogni classe di guasto
%ricavo il numero di classi di guasto

[s,numTclas]=size(D{1,4}); [sizeMBRG,lengthMBRG]=size(MBRG);
diag=zeros(1,numTclas);

tic

for f=1:numTclas

fprintf('\n********** CLASSE DI GUASTO %d **********\n',f);

%Per ogni classe di guasto 'disattivo' nell'MBRD i nodi che hanno Delta~=2
%ottenendo una matrice di celle MD di cui cotruisco il grafo G

[MD,G]=MBRD_mod(D,sizeE,f);

%Per il calcolo dei cicli è stato utilizzato l'Algoritmo di Johnson che,
%però lavora con grafi che non hanno self loop e archi multipli tra due
%nodi, per cui utilizzo la funzione elimina.m che appunto elimina dal grafo
%G i self loop e gli archi multipli

[G,loop]=elimina(G);

%Del grafo G risultante ricavo le componenti fortemente connesse

comp=scc(G); [CFC,Ak]=cfc(comp,G);

%Applicazione dell'Algoritmo di Johnson

[CICLI]=algoritmo_Johnson1(G,Ak,loop);
```

```
%************************************************************************
%INIZIO DELLA VERIFICA DELLA DIAGNOSTICABILITA'
%************************************************************************

[sizeC,lengthC]=size(CICLI);

if (sizeC==0)

    %se l'MBRD non presenta cicli incerti è possibile affermare che il
    %sistema è diagnosticabile in riferimento alla classe di guasto in
    %esame

    fprintf('\n Il sistema è diagnosticabile in riferimento alla classe di
    guasto %d in quanto non sono presenti cicli incerti nel MBRD\n\n',f)

    diag(1,f)=1;
else
   %se sono presenti dei cicli incerti è necessario valutare se essi sono
   %indeterminati

   indet=zeros(1,sizeC);
   for c=1:sizeC
   %per ogni ciclo relativo alla classe di guasto f, devo ricavare la parola
   %o le parole corrispondenti e verificare se il ciclo è indeterminato
   WORD=[''];
   [sizeciclo,lengthciclo]=size(CICLI{c,1});
   x=0;

%************************************************************************
%DETERMINAZIONE DELLA PAROLA O DELLE PAROLE ASSOCIATE AL CICLO IN  ESAME
%************************************************************************

       for j=1:lengthciclo-1 %esploro il ciclo
           nodocorrente=CICLI{c,1}(1,j);
           nodosuccessivo=CICLI{c,1}(1,j+1);
           [sizeDi,numdiarchi]=size(D{nodocorrente,3});
           a=[];
           count=1;

             for jj=1:numdiarchi
             %è possibile che dal 'nodocorrente' si raggiunga il
             %'nodosuccessivo' con lo scatto di più etichette(archi multipli)
               if (size(D{nodocorrente,3}{jj})~=0)
               n=find(D{nodocorrente,3}{jj}{2}==nodosuccessivo);
               if(size(n)~=0)
               a=[a jj];
               end
               end
             end

             [sizeW,lengthW]=size(WORD);
             %al primo passo WORD avrà dimensione nulla
             if(x==0)
               for jjj=1:length(a)
               etichetta=D{nodocorrente,3}{a(jjj)}{1};
               WORD(jjj,lengthW+1)=etichetta;
               end
```

```
              x=1;
             else
             [sizeW,lengthW]=size(WORD);
             for jjj=1:length(a)
             %nel caso in cui ci siano archi multipli devo ricavare i
             %valori delle diverse etichette
             etichetta=D{nodocorrente,3}{a(jjj)}{1};
             for j=1:sizeW
             %aggiorno tutti gli elementi presenti in WORD con la nuova
             %etichetta
             WORD(count,1:lengthW+1)=[WORD(j,1:lengthW), etichetta];
             count=count+1;
             end
             end
             end
       end

%Una volta determinata la parola o le parole corrispondenti al ciclo
%in esame devo ricavare le matrici di cicli N e F.

%*************************************************************************
%COSTRUZIONE DELLA MATRICE DI CICLI N
%*************************************************************************

%La matrice di celle N ha una sola riga e un numero di colonne pari al
%numero di nodi del ciclo considerato meno una.Ogni colonna mostra quali
%sono i nodi(dell'MBRG) che hanno h=N per la classe di guasto in esame

      N=[];
      [sizeDi,lengthDi]=size(D{1,2});
      p=lengthDi-1-numTclas;
      for j=1:lengthciclo-1
      [sizeDi,lengthDi]=size(D{CICLI{c,1}(1,j),2});
      d=find(D{CICLI{c,1}(1,j),2}(1:sizeDi,1+p+f)==-1);
        for jj=1:length(d)
            N{1,j}(jj,1)=D{CICLI{c,1}(1,j),2}(d(jj),1);
        end
      end


     for i=1:length(N)
         x=(N{1,i})';
         s=sort(x);
         N{1,i}=s';
     end


%Una volta costruita la matrice di celle N vengono valutate tutte le
%possibili combinazioni di nodi, in modo da ottenere i cicli N, la cui
%esistenza andrà poi verificata nel MBRG

     cicli_N=[];
     [sizeN,lengthN]=size(N);
     xx=0;

     for i=1:lengthN
      [sizeN,lengthN]=size(N);
      count=0;
         [sizeNi,lengthNi]=size(N{1,i});
```

```
         if(xx==0)
             for j=1:sizeNi
             cicli_N(j,1)=N{1,i}(j,1);
             end
             xx=1;
         else
             [sizeM_N,lengthM_N]=size(cicli_N);

             v=(N{1,i}(:,1))';
             for jj=1:length(v)
                 nodo=v(jj);
              for jjj=1:sizeM_N
               cicli_N(count+1,1:lengthM_N+1)=[cicli_N(jjj,1:lengthM_N) nodo];
               count=count+1;
               end
             end

         end
     end

   [sizeM_N,lengthM_N]=size(cicli_N);
             for r=1:sizeM_N
                 cicli_N(r,lengthM_N+1)=cicli_N(r,1:1);
             end

%Elimino gli eventuali doppioni

[sizeM_N,lengthM_N]=size(cicli_N);
        for i=1:sizeM_N
            for j=1:sizeM_N
                if (i~=j &(cicli_N(i,:)==cicli_N(j,:)))
                    cicli_N(j,:)=-3;
                end
            end
        end
        temp=[];
        countM=0;
        for j=1:sizeM_N
            if (cicli_N(j,:)~=-3)
                countM=countM+1;
                temp(countM,:)=cicli_N(j,:);
            end
        end

    cicli_N=temp;


%************************************************************************
%COSTRUZIONE DELLA MATRICE DI CICLI F
%************************************************************************

%La matrice di celle F ha una sola riga e un numero di colonne pari al
%numero di nodi del ciclo considerato meno una.Ogni colonna mostra quali
%sono i nodi(dell'MBRG) che hanno h=F per la classe di guasto in esame

      F=[];
      [sizeDi,lengthDi]=size(D{1,2});
      p=lengthDi-1-numTclas;
      for j=1:lengthciclo-1
```

```
      [sizeDi,lengthDi]=size(D{CICLI{c,1}(1,j),2});
      d=find(D{CICLI{c,1}(1,j),2}(1:sizeDi,1+p+f)==-2);
        for jj=1:length(d)
            F{1,j}(jj,1)=D{CICLI{c,1}(1,j),2}(d(jj),1);
        end
      end

     for i=1:length(F)
         x=(F{1,i})';
         s=sort(x);
         F{1,i}=s';
     end


%Una volta costruita la matrice di celle N vengono valutate tutte le
%possibili combinazioni di nodi, in modo da ottenere i cicli N, la cui
%esistenza andrà poi verificata nellMBRG

     cicli_F=[];
     [sizeF,lengthF]=size(F);
     xx=0;

     for i=1:lengthF
      [sizeF,lengthF]=size(F);
      count=0;
         [sizeFi,lengthFi]=size(F{1,i});
         if(xx==0)
             for j=1:sizeFi
             cicli_F(j,1)=F{1,i}(j,1);
             end
             xx=1;
         else
             [sizeM_F,lengthM_F]=size(cicli_F);

             v=(F{1,i}(:,1))';
             for jj=1:length(v)
                 nodo=v(jj);
              for jjj=1:sizeM_F
               cicli_F(count+1,1:lengthM_F+1)=[cicli_F(jjj,1:lengthM_F) nodo];
               count=count+1;
               end
             end

         end
     end

     [sizeM_F,lengthM_F]=size(cicli_F);
             for r=1:sizeM_F
                 cicli_F(r,lengthM_F+1)=cicli_F(r,1:1);
             end

%Elimino gli eventuali doppioni

[sizeM_F,lengthM_F]=size(cicli_F);
        for i=1:sizeM_F
            for j=1:sizeM_F
                if (i~=j &(cicli_F(i,:)==cicli_F(j,:)))
                    cicli_F(j,:)=-3;
                end
```

```
            end
        end
        temp=[];
        countM=0;
        for j=1:sizeM_F
            if (cicli_F(j,:)~=-3)
                countM=countM+1;
                temp(countM,:)=cicli_F(j,:);
            end
        end

    cicli_F=temp;


%Una volta ricavate le matrici di cicli N e F corrispondenti a nodi
%dell'MBRG bisogna verificare se esiste almeno un ciclo F e almeno un ciclo
%N nell'MBRG, in tal caso significherebbe che il ciclo trovato nell'MBRD è
%indeterminato e quindi il sistema sarebbe non diagnosticabile per la
%classe di guasto in esame.

[sizeW,lengthW]=size(WORD);
for w=1:sizeW %ad ogni ciclo potrebbe essere associata più di una parola

cicli_F_trovati=[]; cicli_N_trovati=[]; countF=0; countN=0;
[sizeF,lengthF]=size(cicli_F);

    for j=1:sizeF
        ciclo=cicli_F(j,:);
        cl_F=zeros(1,lengthF-1);
        for jj=1:lengthF-1
        scatto_etichetta=0;
        nodocorrente=ciclo(1,jj);
        nodosuccessivo=ciclo(1,jj+1);
        etichetta=WORD(w,jj);

        [sizeM_8,lengthM_8]=size(MBRG{nodocorrente,8});

        for n=1:sizeM_8
            for nn=1:sizeE
             if(size(MBRG{nodocorrente,8}{n,nn})~=0)
              if((MBRG{nodocorrente,8}{n,nn}{1,1}==etichetta) &
              (MBRG{nodocorrente,8}{n,nn}{1,3}{1,1}==nodosuccessivo))
                    scatto_etichetta=1;
                    cl_F(1,jj)=1;
                    break
              elseif((MBRG{nodocorrente,8}{n,nn}{1,1}==etichetta) &
              (MBRG{nodocorrente,8}{n,nn}{1,3}{1,1}~=nodosuccessivo))
                    scatto_etichetta=2;
                    nodo_ragg=MBRG{nodocorrente,8}{n,nn}{1,3}{1,1};

%Nel caso in cui scatto_etichetta=2 significa che dal nodo in esame è
%scattata l'etichetta ma mi porta ad un nodo diverso da quello voluto, a
%questo punto bisogna verificare se da questo nodo posso raggiungere il
%nodo desisderato con lo scatto di uno o più guasti

                    num_nodo=zeros(1,sizeMBRG);
                    num_nodo(1,nodo_ragg)=1;
                    while(find(num_nodo==1)~=0)
```

```
      %devo verificare le transizioni di guasto abilitate da quel nodo
                  v=find(num_nodo==1);
                  %vettore contenente gli indici di num_nodo=1

                  vv=find(MBRG{v(1,1),4}(1,no+1:no+nf)==1);

      %vettore contenente gli indici delle transizioni di guasto
      %abilitate dal nodo

                  num_nodo(1,v(1,1))=2; %nodo esplorato
                  [rvv,cvv]=size(vv);
                  for i=1:cvv
                   nodo=MBRG{v(1,1),5}{1,no+vv(i)}{1,1}{1,1};

      %mi restituisce il numero del nodo raggiunto con lo scatto
      %della transizione di guasto no+vv(i)

                   if(nodo==nodosuccessivo)
                     cl_F(1,jj)=1;
                   break
                   elseif(num_nodo(1,nodo)==0)
                     num_nodo(1,nodo)=1;
                   end
                  end

                  if(cl_F(1,jj)==1)
                   break
                  end

                  end
              end
              end
          end
          if (cl_F(1,jj)==1)
              break
          end
       end

       if(scatto_etichetta==0)
         num_nodo_a=zeros(1,sizeMBRG);
         num_nodo_a(1,nodocorrente)=1;
         while(find(num_nodo_a==1)~=0)

   %devo verificare le transizioni di guasto abilitate dal nodo

         v_a=find(num_nodo_a==1);
         vv_a=find(MBRG{v_a(1,1),4}(1,no+1:no+nf)==1);
         num_nodo_a(1,v_a(1,1))=2;
         [rvv_a,cvv_a]=size(vv_a);
         for i=1:cvv_a
             nodo_a=MBRG{v_a(1,1),5}{1,no+vv_a(i)}{1,1}{1,1};

   %a questo punto devo verificare se da questo nodo può scattare
   %l'etichetta e mi porta al nodo successivo oppure se scatta l'etichetta
   %e mi porta  ad un nodo diverso e in tal caso se da qui con lo scatto di
   %una o più trans. di guasto raggiungo il nodo successivo

         [sizeM_8,lengthM_8]=size(MBRG{nodo_a,8});
           for n=1:sizeM_8
```

```
                for nn=1:sizeE
                    if(size(MBRG{nodo_a,8}{n,nn})~=0)
                        if((MBRG{nodo_a,8}{n,nn}{1,1}==etichetta) &
                        (MBRG{nodo_a,8}{n,nn}{1,3}{1,1}==nodosuccessivo))

                        cl_F(1,jj)=1;
                        break
                        elseif((MBRG{nodo_a,8}{n,nn}{1,1}==etichetta) &
                        (MBRG{nodo_a,8}{n,nn}{1,3}{1,1}~=nodosuccessivo))

                          nodo_ragg=MBRG{nodo_a,8}{n,nn}{1,3}{1,1};
                          num_nodo_b=zeros(1,sizeMBRG);
                          num_nodo_b(1,nodo_ragg)=1;
                          while(find(num_nodo_b==1)~=0)
                          v_b=find(num_nodo_b==1);
                          vv_b=find(MBRG{v_b(1,1),4}(1,no+1:no+nf)==1);
                          num_nodo_b(1,v_b(1,1))=2;
                          [rvv_b,cvv_b]=size(vv_b);
                           for ii=1:cvv_b
                             nodo_b=MBRG{v_b(1,1),5}{1,no+vv_b(i)}{1,1}{1,1};
                             if(nodo_b==nodosuccessivo)
                                cl_F(1,jj)=1;
                                break
                             elseif(num_nodo_b(1,nodo_b)==0)
                                num_nodo_b(1,nodo_b)=1;
                             end
                           end
                           if(cl_F(1,jj)==1)
                            break
                           end
                          end
                       end
                    end
                    if(cl_F(1,jj)==1)
                        break
                    end
                end
                if(cl_F(1,jj)==1)
                   break
                end
            end
          if(cl_F(1,jj)==1)
              break
          elseif(num_nodo_a(1,nodo_a)==0)
              num_nodo_a(1,nodo_a)=1;
          end
          end
         if(cl_F(1,jj)==1)
             break
         end
        end
end
    end

        if(cl_F(1,:)==ones(1,lengthF-1))
         %Ciclo F trovato nello MBRG
            break
        end
   end
```

```
  if(cl_F(1,:)~=ones(1,lengthF-1))
    break
   else
 [sizeN,lengthN]=size(cicli_N);
    for j=1:sizeN
    ciclo=cicli_N(j,:);
    cl_N=zeros(1,lengthN-1);
     for jj=1:lengthN-1
     scatto_etichetta=0;
     nodocorrente=ciclo(1,jj);
     nodosuccessivo=ciclo(1,jj+1);
     etichetta=WORD(w,jj);

     [sizeM_8,lengthM_8]=size(MBRG{nodocorrente,8});

        for n=1:sizeM_8
            for nn=1:sizeE
                if(size(MBRG{nodocorrente,8}{n,nn})~=0)
                if((MBRG{nodocorrente,8}{n,nn}{1,1}==etichetta) &
                (MBRG{nodocorrente,8}{n,nn}{1,3}{1,1}==nodosuccessivo))

                    scatto_etichetta=1;
                    cl_N(1,jj)=1;
                    break
                elseif((MBRG{nodocorrente,8}{n,nn}{1,1}==etichetta) &
                (MBRG{nodocorrente,8}{n,nn}{1,3}{1,1}~=nodosuccessivo))

                    scatto_etichetta=2;
                    nodo_ragg=MBRG{nodocorrente,8}{n,nn}{1,3}{1,1};

%Nel caso in cui scatto_etichetta=2 significa che dal nodo in esame è
%scattata l'etichetta ma mi porta ad un nodo diverso da quello voluto, a
%questo punto bisogna verificare se da questo nodo posso raggiungere il
%nodo desisderato con lo scatto di uno o più guasti

                    num_nodo=zeros(1,sizeMBRG);
                    num_nodo(1,nodo_ragg)=1;
                    while(find(num_nodo==1)~=0)

      %devo verificare le transizioni di guasto abilitate da quel nodo
                    v=find(num_nodo==1);
                    %vettore contenente gli indici di num_nodo=1

                    vv=find(MBRG{v(1,1),4}(1,no+1:no+nf)==1);

      %vettore contenente gli indici delle transizioni di guasto
      %abilitate dal nodo

                    num_nodo(1,v(1,1))=2; %nodo esplorato
                    [rvv,cvv]=size(vv);
                    for i=1:cvv
                    nodo=MBRG{v(1,1),5}{1,no+vv(i)}{1,1}{1,1};

      %mi restituisce il numero del nodo raggiunto con lo scatto
      %della transizione di guasto no+vv(i)

                     if(nodo==nodosuccessivo)
                       cl_N(1,jj)=1;
```

```
                 break
                 elseif(num_nodo(1,nodo)==0)
                   num_nodo(1,nodo)=1;
                 end
               end

               if(cl_N(1,jj)==1)
                 break
               end

               end
           end
           end
       end
       if (cl_N(1,jj)==1)
           break
       end
   end

   if(scatto_etichetta==0)
     num_nodo_a=zeros(1,sizeMBRG);
     num_nodo_a(1,nodocorrente)=1;
     while(find(num_nodo_a==1)~=0)
%devo verificare le transizioni di guasto abilitate dal nodo
     v_a=find(num_nodo_a==1);
     vv_a=find(MBRG{v_a(1,1),4}(1,no+1:no+nf)==1);
     num_nodo_a(1,v_a(1,1))=2;
     [rvv_a,cvv_a]=size(vv_a);
     for i=1:cvv_a
         nodo_a=MBRG{v_a(1,1),5}{1,no+vv_a(i)}{1,1}{1,1};

%a questo punto devo verificare se da questo nodo può scattare
%l'etichetta e mi porta al nodo successivo oppure se scatta
%l'etichetta e mi porta  ad un nodo diverso e in tal caso se da
%qui con lo scatto di una o più trans. di guasto raggiungo il
% nodo successivo

     [sizeM_8,lengthM_8]=size(MBRG{nodo_a,8});
       for n=1:sizeM_8
           for nn=1:sizeE
               if(size(MBRG{nodo_a,8}{n,nn})~=0)
                   if((MBRG{nodo_a,8}{n,nn}{1,1}==etichetta) &
                   (MBRG{nodo_a,8}{n,nn}{1,3}{1,1}==nodosuccessivo))

               cl_N(1,jj)=1;
               break
                  elseif((MBRG{nodo_a,8}{n,nn}{1,1}==etichetta) &
                  (MBRG{nodo_a,8}{n,nn}{1,3}{1,1}~=nodosuccessivo))

                  nodo_ragg=MBRG{nodo_a,8}{n,nn}{1,3}{1,1};
                  num_nodo_b=zeros(1,sizeMBRG);
                  num_nodo_b(1,nodo_ragg)=1;

                    while(find(num_nodo_b==1)~=0)
                    v_b=find(num_nodo_b==1);
                    vv_b=find(MBRG{v_b(1,1),4}(1,no+1:no+nf)==1);
                    num_nodo_b(1,v_b(1,1))=2;
                    [rvv_b,cvv_b]=size(vv_b);
                      for ii=1:cvv_b
```

```
                                nodo_b=MBRG{v_b(1,1),5}{1,no+vv_b(i)}{1,1}{1,1};
                                if(nodo_b==nodosuccessivo)
                                   cl_N(1,jj)=1;
                                   break
                                elseif(num_nodo_b(1,nodo_b)==0)
                                   num_nodo_b(1,nodo_b)=1;
                                end
                              end

                              if(cl_N(1,jj)==1)
                                break
                              end
                            end
                        end

                        if(cl_N(1,jj)==1)
                            break
                        end
                    end

                    if(cl_N(1,jj)==1)
                       break
                    end
                end

            if(cl_N(1,jj)==1)
                break
            elseif(num_nodo_a(1,nodo_a)==0)
                num_nodo_a(1,nodo_a)=1;
            end
            end

           if(cl_N(1,jj)==1)
               break
           end
          end
 end
     end

         if(cl_N(1,:)==ones(1,lengthN-1))
             break
         end
    end
  end

    if(cl_F(1,:)==ones(1,lengthF-1) & cl_N(1,:)==ones(1,lengthN-1))
        indet(1,c)=1;
        break
    end
end
   if(cl_F(1,:)==ones(1,lengthF-1) & cl_N(1,:)==ones(1,lengthN-1))
        break
   end

   end

    indet;
    if(find(indet==1)~=0)
```

```
     fprintf('\nIl sistema è non diagnosticabile in riferimento alla classe
     di guasto %d per la presenza di cicli indeterminati\n\n',f)

    else
     fprintf('\nIl sistema è diagnosticabile in riferimento alla classe
     di guasto %d in quanto non sono presenti cicli indeterminati\n\n',f)
     diag(1,f)=1;
    end

end

%Analizzo una classe di guasto per volta
% disp('[PREMI UN TASTO PER CONTINUARE]')
% pause
end

fprintf('**************************************\n')

if(diag==ones(1,numTclas))
    fprintf('Il sistema è diagnosticabile\n\n')
else
    fprintf('Il sistema non è diagnosticabile\n\n')
end

fprintf('**************************************\n')

toc

end
```

Bibliografia

[1] L. Osterwell A. Ehrenfeucht, L. Fosdick. An algorithm for finding the elementary circuits of a directed graph. *Tech. Rep. CU-CS-024-23, Dept. of Computer Sci., Univ. of Colorado, Boulder*, 1973. [cited at p. 5]

[2] A. Di Febbraro A. Giua. *Sistemi ad eventi discreti*. The McGraw-Hill Companies, 2002. [cited at p. 9]

[3] 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. [cited at p. 2]

[4] 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. [cited at p. 2]

[5] 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. [cited at p. 2]

[6] 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. [cited at p. 2]

[7] 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. [cited at p. 2]

[8] M.P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu. Diagnosability analysis of unbounded Petri nets. In *Proc. 48th IEEE Conf. on Decision and Control*, December 2009. [cited at p. 3]

[9] M.P. Cabasino, A. Giua, and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 2008. Preliminary accepted. [cited at p. 2, 3, 23, 26, 27, 28]

[10] 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. [cited at p. 2, 3, 5, 6, 7, 21, 28, 31, 55, 83, 87, 88]

[11] 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. [cited at p. 2, 23, 24, 25, 26]

[12] S.L. Chung. Diagnosing pn-based models with partial observable transitions. *International Journal of Computer Integrated Manufacturing*, 12 (2):158–169, 2005. [cited at p. 3]

[13] 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. [cited at p. 2]

[14] 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. [cited at p. 2]

[15] 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. [cited at p. 3]

[16] 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. [cited at p. 2]

[17] 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. [cited at p. 2]

[18] Donald B. Johnson. Finding all the elementary circuit of a directed graph. *SIAM J. Comput.*, 4:77–84, 1975. [cited at p. 5, 6, 39, 68]

[19] S. Lafortune. Executables of the umdes-lib software library for solaris, linux, mac and windows are publicly available. *http://www.eecs.umich.edu/umdes/toolboxes.html*. [cited at p. 4]

[20] J. Lunze and J. Schroder. Sensor and actuator fault diagnosis of systems with discrete inputs and outputs. 34(3):1096–1107, April 2004. [cited at p. 2]

[21] C.A. Petri. *Kommunication mit Automaten*. PhD thesis, Institut fur Instrumentelle Mathematik, Schriften des IIM, No. 3, Bonn, Germany, 1962. [cited at p. 9]

[22] Marco Pocci. Un toolbox per la diagnosticabilità di reti posto/transizione. Master's thesis, Dep. Electric and Electronic Engineering, University of Cagliari, Cagliari, Italy, 2009. (In Italian). [cited at p. 71, 86]

[23] M. Sampath. *A Discrete Event Systems Approach to Failure Diagnosis*. PhD thesis, The University of Michigan, Ann Arbor, Michigan, 1995. [cited at p. 1]

[24] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Trans. on Automatic Control*, 43(7):908–929, July 1998. [cited at p. 2]

[25] 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. [cited at p. 3]

[26] 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. [cited at p. 3]

[27] Robert Tarjan. Enumeration of the elementary circuit of a directed graph. *SIAM J. Comput.*, 2:211–216, 1972. [cited at p. 5, 39]

[28] James C. Tiernan. An efficient search algorithm to find the elementary circuits of a graph. *Comm. ACM*, 13:722–726, 1970. [cited at p. 5, 39]

[29] 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. [cited at p. 2, 3]

[30] H. Weinblatt. A new search algorithm for finding the simple cycles of a finite directed graph. *J. Assoc. Comput. Mach.*, 19:43–56, 1972. [cited at p. 5]

[31] Y. Wen and M. Jeng. Diagnosability analysis based on T-invariants of Petri nets. In *Networking, Sensing and Control, 2005. Proceedings, 2005 IEEE.*, pages 371–376, March 2005. [cited at p. 3]

[32] 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. [cited at p. 2]