UNIVERSITÀ DEGLI STUDI DEL SANNIO

Dipartimento di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea

MODEL CHECKING PROBABILISTICO PER L'INFERENZA DI RETI GENICHE

Relatore:

Prof.ssa ANTONELLA SANTONE

Correlatore:

Dott. GIUSEPPE DE RUVO

Laureanda:

NARDONE VITTORIA

ANNO ACCADEMICO 2013-2014

Un sentito ringraziamento per la costante fiducia e il grande affetto va ai miei genitori, che, con il loro incrollabile sostegno morale ed economico, mi hanno permesso di raggiungere questo secondo importantissimo traguardo.

Ringrazio anche i miei fratelli, Orlando e Nicola, che in tutti questi anni non mi hanno mai fatto mancare il sorriso e i momenti di conforto.

Desidero ringraziare la professoressa Antonella Santone, per i consigli, la competenza, la presenza, la pazienza e la disponibilitá mostrata nei miei confronti.

Ringrazio l'Ing. Giuseppe De Ruvo per la sua grande professionalitá e per sui consigli, che sono risultati sempre utilissimi.

Un grandissimo ringraziamento va a tutti i miei amici, per essermi stati sempre vicino, per avermi sopportato anche nei momenti di gran difficoltá e per avermi sempre dato una parola d'incoraggiamento.

Ringrazio le coinquiline di "'Casa Tarantino"' (attuali ed ex), con cui ho condiviso gioie, dolori, frustrazioni e soddisfazioni non soltanto della tesi ma di tutta questa lunga avventura univeristaria.

In particolar modo ringrazio Fiorella, una grandissima amica che ho conosciuto, forse, troppo tardi, ma che sento di conoscere da sempre.

Un immenso grazie va anche alla mia amica e compagna di avventure di sempre, Maria Carmela. Con lei ho affrontato questo lungo percorso universitario e condiviso tutto, dalle gioie ai dolori.

Concludo infine con un augurio rivolto a tutte quelle persone che mi voglio bene quanto gliene voglio io:

Puntate sempre alla luna... Male che vada avrete vagato tra le stelle...

$\lambda\mu\sigma$ **GRAZIE DI TUTTO** $\lambda\mu\sigma$

Quello che sono lo devo a voi ...

Indice

Bibliografia **63**

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

La modellizzazione di sistemi biologici è un passo cruciale nella comprensione del loro funzionamento e nelle applicazioni quali il controllo di tali sistemi e l'ingegnerizzazione di nuove funzioni biologiche. Spesso le dinamiche dei sistemi biologici non sono completamente note, pertanto si pone il problema di identificare modelli matematici attraverso cui ricavare struttura e/o parametri non noti a partire da dati sperimentali.

Le moderne biotecnologie, quali microarray e next generation sequencing, permettono di misurare simultaneamente l'espressione di migliaia di geni in un dato istante temporale e in diverse condizioni sperimentali. A partire da tali misure, uno degli obiettivi primari della Systems Biology consiste nel fornire metodi per ricostruire la topologia della rete di interazioni (funzionali) tra i geni, ossia i circuiti di regolazione di gran parte delle funzioni cellulari.

In questo lavoro di tesi, dopo aver inquadrato il problema dal punto di vista biologico e fornito una panoramica dei principali approcci matematici utilizzati per l'inferenza (o reverse-engineering), ci occuperemo della definizione di una metodologia per l'identificazione di reti genetiche.

La metodologia in questione parte dalla definizione di un modello proba-

bilistico che simula il comportamento della rete mediante l'utilizzo di metodi formali. A valle della creazione del modello sono state definite delle proprietà al fine di effettuare una verifica formale dello stesso; questa verifica è stata basata sul model checking. Il model checker utilizzato per la verifica formale è il PRISM, che è un model checker probabilistico per la modellazione e l'analisi dei sistemi che mostrano un comportamento casuale o probabilistico.

Le proprietà, scelte per la verifica formale, sono state ricavate a partire da un'analisi preliminare dei dati a disposizione e ispirandosi a quanto già presente in letteratura [5].

I risultati ottenuti dalla verifica formale sono elaborati da un algoritmo creato ad hoc, il quale restituisce, sotto forma di "archi", le relazioni che intercorrono tra i geni della rete, ovvero il modo in cui questi ultimi interagiscono tra di loro.

Per ciascuna relazione individuata l'algoritmo è in grado di rilevare sia la direzione che il segno, ovvero inibizione o attivazione del gene, mentre la maggior parte dei metodi di letteratura sono limitati a rapporti non orientati e/o senza segno.

Al fine di validare, in termini di veridicità, le relazioni individuate è stato effettuato un confronto tra queste ultime e le reali relazioni esistenti tra i geni. Suddetto confronto è stato valutato in modo empirico in termini di precision e recall. Infine i risultati empirici sono stati rapportati a quelli presenti in letteratura con l'obiettivo di verificare se la metodologia proposta introduce delle migliorie in termini di accuratezza e precisione dei risultati.

1.1 Struttura della Tesi

La tesi è strutturata come segue:

- Il capitolo 1 contiene l'introduzione al lavoro di tesi svolto durante l'attività di tirocinio.
- Nel capitolo 2 vengono introdotti alcuni concetti biologici atti a facilitare la comprensione del lavoro svolto. Successivamente viene descritto lo stato dell'arte.
- Il capitolo 3 descrive la metodoligia e il tool Prism utilizzati nel lavoro di tesi.
- Il capitolo 4 dettaglia il modello e le proprietà prodotte. Inoltre descrive l'algoritmo per l'individuazione degli archi della rete genica.
- Il capitolo 5 descrive le prestazioni dell'algotirmo e le confronta con quelle del FormalM.
- Nel capitolo 6 sono riportate le conclusioni e gli eventuali sviluppi futuri.

Capitolo 2

Stato dell'Arte

In letteratura sono stati proposti diversi metodi di reverse engineering per inferire sulle GNR (Genetic Networks Regulation) a partire dai dati dinamici di espressione genica di microarrays. Il termine reverse engineering indica l'insieme dei metodi utili a ricostruire la complessa rete di regolazione e controllo dall'output dinamico del sistema osservato. Data la complessità del sistema analizzato, gli approcci di reverse engineering per lo studio dell'interazione tra geni, proteine ed altri metaboliti, è in genere limitato a piccole parti della rete di regolazione. Prima di effettuare una breve panoramica su quelli che sono i metodi presenti in letteratura è necessario focalizzare l'attenzione su alcuni concetti prettamente biologici.

2.1 Excursus Biologico

In questo paragrafo verranno accennati due concetti chiave: espressione genica e reti di regolazione genica; definiti per facilitare la comprensione dell'intero lavoro di tesi realizzato. Infatti a partire da dati in ingresso rappresentativi di un'espressione genica è stata ricavata la rispettiva rete di regolazione genica.

2.1.1 Espressione Genica

In ogni cellula somatica di ogni organismo eucariote è contenuto tutto il genoma, ovvero tutta l'informazione necessaria all'organismo stesso per poter sopravvivere, codificata all'interno del DNA o codice genetico. L'informazione genica specifica la natura e le proprietà delle proteine e delle molecole funzionali che l'organismo è in grado di produrre.

In biologia molecolare, con il termine **espressione genica** si intende il processo attraverso cui l'informazione contenuta in un gene (costituita di DNA) viene convertita in una macromolecola funzionale (tipicamente una proteina). In particolare, l'espressione genica diversificata rappresenta la differenziazione, lo sviluppo e l'attività cellulare nonché la risposta e l'adattamento agli stimoli esterni. Le cellule non contengono solo le istruzioni per la codifica delle proteine, ma anche l'informazione relativa alle condizioni in cui le proteine devono essere sintetizzate.

Questa informazione si esplica attraverso meccanismi di regolazione e controllo molto complessi, i cui due passaggi principali sono la trascrizione, durante la quale il DNA è trascritto in mRNA, e la traduzione, durante la quale l'mRNA (RNA messaggero) è tradotto in proteina.

Un altro concetto da chiarire riguarda il **gene expression profiling** che misura l'attività (espressione) di migliaia di geni alla volta, per creare un'immagine globale della funzione cellulare. Questi profili possono, ad esempio, mostrare come le cellule reagiscono ad un particolare trattamento.

2.1.2 Reti di Regolazione Genica

Le reti di regolazione genica o **GNR** (dall'inglese **Genetic Networks Regulation**) descrivono le complesse interazioni che influenzano l'espressione genica

e, conseguentemente, il comportamento cellulare. La rete solitamente viene rappresentata mediante l'utilizzo di un grafo. Un **grafo** è un insieme di elementi detti **nodi** che possono essere collegati fra loro da linee chiamate **archi**. Più formalmente, si dice grafo una coppia ordinata $G = (V, E)$ di insiemi, con V insieme dei nodi ed E insieme degli archi, tali che gli elementi di E siano coppie di elementi di V ($E \subseteq V \times V$).

Esistono diverse tipologie di reti in base alla tipologia di archi utilizzati, si parla infatti di:

- reti orientate o non orientate (se gli archi hanno o meno una direzione fissata);
- reti pesate o non pesate (se ad ogni arco viene o meno assegnato un peso).

Si parla di reti in molteplici contesti differenti quali matematica, fisica, biologia, sociologia ed economia. Tutte le reti sono simili come struttura in quanto sono sempre costituite da nodi e archi e possono essere analizzate con gli stessi criteri (teoria dei grafi), ma differiscono per il significato che nodi e archi assumono nei diversi casi.

Negli ultimi anni si sta manifestando un crescente interesse per quanto riguarda lo studio delle interazioni esistenti tra le varie molecole presenti nelle cellule. L'obiettivo che cercano di raggiungere numerosi studiosi, utilizzando differenti approcci, è quello di ricostruire i meccanismi di regolazione che coinvolgono geni, proteine e metaboliti a diversi livelli, tentando di ricostruire le biochemical networks. In base al livello di dettaglio sul quale si lavora, le reti biochimiche possono essere suddivise nelle seguenti tre categorie:

1. *reti geniche*: rappresentano le relazioni che si instaurano tra geni andando a vedere in che modo il livello di espressione di un gene influisce sul livello di espressione degli altri;

2. *reti proteiche*: rappresentano le interazioni che avvengono tra le proteine quali la formazione di complessi e le modifiche proteiche causate dagli enzimi di signaling (in questo caso si parla anche di reti di signaling);

3. *reti metaboliche*: rappresentano le reazioni chimiche che avvengono tra i metaboliti.

Poiché le tecnologie attuali non permettono di monitorare l'espressione delle proteine, come invece i microarray fanno per i trascritti, si è focalizzata l'attenzione sull'utilizzo delle reti geniche.

Spesso le reti geniche vengono rappresentate graficamente utilizzando cerchi o punti per identificare i nodi, e linee o frecce per identificare gli archi. La direzione delle interazioni, se presente, è indicata dall'orientazione degli archi stessi, che quindi sono rappresentati con una freccia. I pesi degli archi, invece, dipendono dal modello che ipotizziamo per la rete ma solitamente indicano l'intensità del controllo. Risulta evidente che tutti gli archi con peso nullo sono regolazioni inesistenti e viceversa, quindi è possibile non rappresentarli graficamente.

Risulta evidente che l'elemento di maggior interesse della rete di regolazione genica così definita sono gli archi, ossia le regolazioni, che intercorrono tra due geni: coppie di nodi (X, Y). Nel caso di rete direzionale, le coppie devono essere ordinate, anteponendo il nodo che controlla a quello controllato, questo corrisponde alla freccia che va da X a Y. Inoltre, se la rete è pesata, ad ogni arco viene associato un peso $w(x, y)$.

Nelle reti reali esistono due tipi di archi:

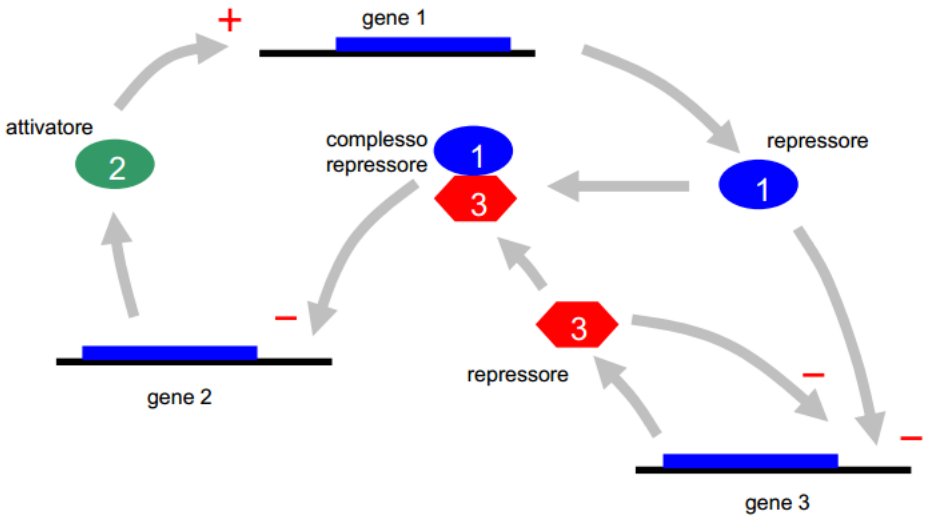

Figura 2.1: Esempio di Rete Genica

- un arco $X \rightarrow Y$ indica che ad un incremento nel tempo dell'espressione di X corrisponde un diretto incremento dell'espressione di Y (arco attivatore);
- un arco $X \dashv Y$ indica che ad un incremento nel tempo dell'espressione di X corrisponde un diretto decremento dell'espressione di Y (arco inibitore).

2.1.3 Reti Sintetiche

Biologicamente la "vera" struttura di una rete è generalmente non individuabile e ciò, chiaramente, ostacola la possibilità di permettere la valutazione di qualsiasi previsione. Per ovviare suddetto problema si è soliti utilizzare reti sintetiche, ovvero reti ottenute mediante un processo di reverse engineering che a partire dagli archi della rete ricava dei possibili relativi dati di espressione. Lo scopo è quello di fornire benchmark e tool per ottenere un collaudo rigoroso di metodi capaci di determinare la regolamentazione di una rete genica.

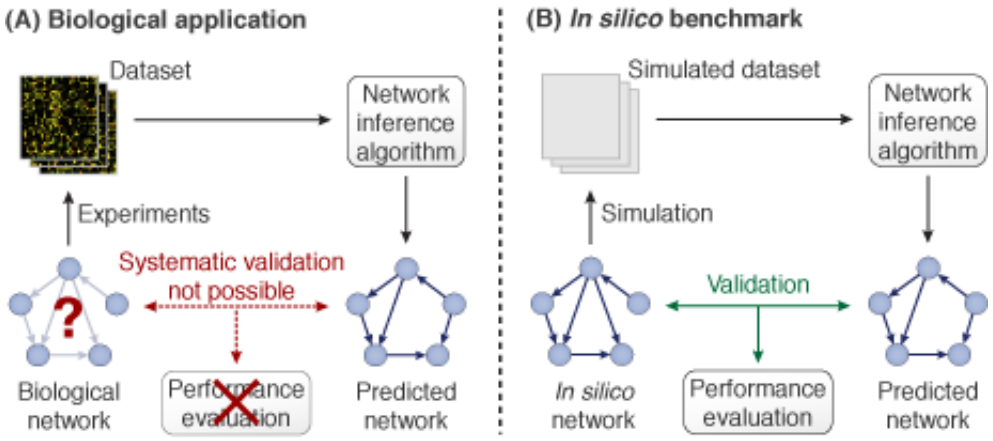

Figura 2.2: Differenza tra rete sintetica e rete biologica

La struttura "artificiale" di una rete è nota a priori, ciò consente di utilizzarla come "oracolo" per valutare le performance dell'algoritmo che ricava la rete genetica a partire dai dati di espressione.

È possibile utilizzare vari tool per generare dei parametri realistici su cui poter testare errori sistematici di algoritmi aventi lo scopo di determinare le regolazioni di una rete. Uno di questi è chiamato GeneNetWeaver (GNW) [20].

GeneNetWeaver (GNW) è uno strumento per la generazione automatica di reti geniche "in silico" allo scopo di consentire benchmark di reverse engineering, dove "in silico" indica nella simulazione anziché in vivo, ossia, in condizioni biologicamente reali.

2.2 Algoritmi Pre-Esistenti

In letteratura sono presenti diverse strategie con l'obiettivo di identificare le relazioni di regolazione genica riducendo lo spazio di ricerca e/o andando ad estendere la quantitá di informazioni indipendenti. Questo aspetto é stato preso in considerazione dal momento che, dal punto di vista computazionale,

la determinazione delle reti di regolazione genica viene visto come un problema indeterminato vista la stragrande quantitá di soluzioni possibili rapportate ad un insieme ridotto di dati indipendenti [10, 2, 3, 11, 21]. In bioinformatica, suddetto problema viene affrontato mediante l'utilizzo di un sistema di equazioni differenziali ordinarie (ODE) mediante le quali si va a modellare il modo in cui le varie componenti di interesse interagiscono. Il modello cosí creato é deterministico e puó essere facilmente simulato, analizzato e risolto a patto peró di avere una dettagliata conoscenza del funzionamento del sistema biologico e allo stesso tempo necessitá di un'elevata potenza computazionale [19].

I modelli computazionali cercano di emulare il fenomeno biologico e spesso adottano euristiche al fine di semplificare il modello stesso. Questi presentano l'enorme vantaggio di poter raprresentare in modo effettivo una varietá di sistemi biologici senza il bisogno di considerare relazioni quantitative precise. Va necessariamente menzionato che i modelli maggiormente utilizzati considerano reti booleane e modelli Bayesiani [14, 22]. In base al modo in cui vengono ricavati possiamodistinguere i modelli computazionali in: deterministici, non deterministici e/o stocastici.

Un altro approccio da considerare nella determinazione del modello é quello basato sul concetto di correlazione tra i geni. In particolar modo le interazioni tra coppie di geni vengono determinate a partire da una misura di similaritá. Queste ultimedevono necessariamente filtrate a partire dalla definizione di uno specifico valore di soglia opportunamente stimato. Il grande vantaggio che l'approccio sovramenzionato ha risiede nel fatto che esso permette di analizzare una grande varietá di informazioni permettendo, quindi, la costruzione dell'intero genoma. Anche se individua in modo alquanto accurato le relazioni, nella maggior parte dei casi non é in grado di ricavarne direzione e/o

segno. Tra gli algoritmi che lavorano secondo questo approccio vanno menzionati:ARACNE [15], TD-Aracne [24], PCA-CMI [23], and CLR [9]. ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks) utilizzando profili di espressioni geniche organizzate in microarray, permette di affrontare la complessitá delle reti quali quelle delle cellule dei mammiferi. Tale metodo utilizza un approccio basato sulla teoria dell'informazione pereliminare la stragrande maggioranza di interazioni indirette tipicamente inferite da un'analisi a coppie.

PCA-CMI, analogamente al caso precedente, ci permette di ricavare le reti di regolazione genica a partire dai dati dell'espressioni geniche. Esso basa il proprio funzionamento sull'individuazione delle dipendenze non lineari e sulla topologia strutturale delle GNRs. In questo approccio la dipendenza tra coppie di metodi viene ricavata a partire dal calcolo del coefficiente CMI. Quest'ultimo, assumendo i dati associati all'espressione genica modellabili mediante una distribuzione gaussiana, viene calcolato considerando le matrici di covarianza tra i geni appartenenti ai microarray di interesse. CLR (Context Likelihood of Relatedness) infine, impiega uno stimatore sensibile al modo in cui i geni sono relazionati ed in modo particolare va a trasformare la stima dell'informazione mutua tra coppie di geni in un valore numerico (z-score), ottenendo dati standardizzati (media nulla e varianza unitaria). Tutti gli algoritmi 'guidati', nel complesso, considerano la ricostruzione delle reti di regolazione genica come un'operazione di apprendimento [17, 4]. Sebbene questi approcci superano, in termini di prestazione, generalmente gli approcci 'non guidati' essi necessitano di un set di dati sul quale poter effettuare l'addestramento del modello (training set) ad esempio un insieme di geni in cui a priori si conosce ilmodo in cui essi interagiscono. Sulla base di queste informazioni é possibile creare una funzione di previsione (modello predittivo) che ci permette di discriminare se

esiste o meno una nuova interazione tra geni [7, 8].

Capitolo 3

Metodi Formali e Model Checking

I metodi formali sono un insieme di tecniche che permettono l'esplorazione esaustiva di tutti i possibili comportamenti di un sistema. L'obiettivo dei metodi formali è quello di esprimere specifiche, in modo preciso e non ambiguo, e definire in modo chiaro quando un'implementazione soddisfa le specifiche. Proprio per questo, solitamente, sono utilizzati per la specifica e per la verifica di sistemi complessi.

Per eseguire l'analisi esaustiva di un sistema viene prodotto un modello formale, ovvero un modello che descrive le caratteristiche salienti del sistema in analisi. Successivamente, su questo modello viene applicata una verifica formale che aiuta a stabilire se un sistema soddisfa una data proprietà o si comporta in accordo a una specifica data. Sono due gli approcci fondamentali per la verifica formale dei sistemi: verifica deduttiva e verifica basata su model checking.

Nel primo caso vengono utilizzati assiomi e regole di inferenza per generare il modello del sistema (sistema formale); le proprietà vengono espresse sotto

forma di teoremi del sistema formale le quali vengono dimostrate mediante l'utilizzo di dimostratori automatici. Suddetto approccio presenta degli svantaggi in quanto è difficile da realizzare e richiede l'intervento di esperti per utilizzare i dimostratori automatici.

Nella verifica basata sul model checking, invece, il sistema è modellato attraverso sistemi a transizione di stato (LTS, Labelled Transition Systems) e viene usata una logica temporale per specificare le proprietà. In questo caso il problema della verifica sul modello viene ridotto ad un problema di ricerca sui grafi. Formalmente il problema è posto così: scelta una proprietà da verificare, espressa come una formula logica temporale p, e un modello M avente stato iniziale s decidere se M, $s| = p$.

Il processo del model checking è composto da tre fasi:

1. **Modellazione**: viene costruito il modello, ovvero viene descritto il comportamento del sistema, usando il linguaggio di descrizione fornito dal Model Checker.

2. **Formalizzazione**: viene formalizzata la proprietà da verificare usando il linguaggio di specifica fornito dal Model Checker.

3. **Esecuzione**: si esegue la verifica della proprietà sul modello.

In questo lavoro di tesi il modello che simula il comportamento reale della rete genica è stato sviluppato mediante l'utilizzo dei metodi formali. La sua validazione è stata effettuata mediante verifica formale basata sul model checking. In particolare il Model Checker utilizzato è Prism, un model checker probabilistico.

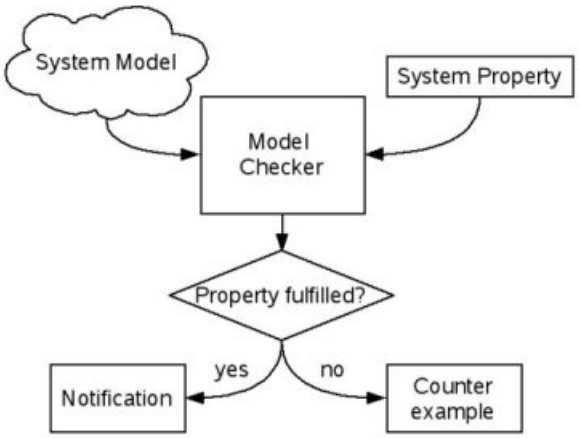

Figura 3.1: Model checking flow

3.1 Prism Model Checker

PRISM [12] è un model checker probabilistico, un tool per la modellazione formale e l'analisi dei sistemi che mostrano un comportamento casuale o probabilistico.

PRISM può costruire e analizzare diversi tipi di modelli probabilistici:

- Catene di Markov a tempo discreto (DTMCs)
- Catene di Markov a tempo continuo (CTMCs)
- Processi decisionali di Markov (MDP)
- Automi probabilistici (PA)
- Automi probabilistici temporizzati (PTA)

I modelli vengono descritti utilizzando il linguaggio di PRISM, un linguaggio semplice, state-based, basato sul formalismo dei moduli reattivi di Alur e Henzinger [1]. PRISM fornisce il supporto per l'analisi automatizzata di una

vasta gamma di proprietà quantitative di questi modelli. Lo sviluppo di PRISM è svolto principalmente presso l'Università di Birmingham e l'Università di Oxford. Il tool è un software open-source, rilasciato sotto GNU General Public License.

3.1.1 Il Linguaggio di Prism

Le componenti fondamentali del linguaggio PRISM sono moduli e variabili. Un modello, infatti, è composto da un numero variabile di moduli che possono interagire tra loro. Un modulo contiene al suo interno una serie di variabili locali. I valori di queste variabili in un dato istante temporale costituiscono lo stato locale del modulo. Lo stato globale del modello completo, invece, viene determinato a partire dallo stato locale di tutti i moduli.

I Moduli

Ogni modulo in Prism rappresenta un processo, ed è specificato come segue:

module *nome_modulo* **endmodule**

La definizione di un modulo contiene due parti: le sue variabili e i relativi comandi. Le variabili descrivono i possibili stati in cui si può trovare il modulo; i comandi descrivono il suo comportamento, ovvero il modo in cui lo stato cambia nel tempo. Attualmente, PRISM supporta solo alcuni tipi semplici di variabili: intere e booleane.

Una possibile dichiarazione di variabile potrebbe essere la seguente:

$$x : [0..2]\ init\ 0;$$

questa dichiarazione ci dice che il modulo ha una variabile interna x che può assumere un valore compreso nell'intervallo specificato ([0..2]) con estre-

mi compresi. L'identificatore *init* specifica il valore iniziale della variabile e quindi lo stato iniziale del modulo. Questa specificazione può anche essere omessa, in questo caso lo stato iniziale assume, di default, il valore più piccolo dell'intervallo, nel caso di variabile intera, oppure falso, in caso di variabile booleana.

In base a quanto detto le variabili definiscono il solo stato del modulo. Per poterne definire il comportamento è necessario inserire nel modulo dei comandi. Ogni comando comprende una guardia e uno o più aggiornamenti. Le condizioni di guardia possono essere sia semplici che multiple, ovvero possono contenere la composizione di più variabili in and oppure in or tra di loro. Le guardie possono contenere vincoli sia su variabili interne al modulo sia su qualsiasi altra variabile esterna specificata in un altro modulo, ovvero il comportamento di un modulo può dipendere anche dallo stato di un altro modulo. Gli aggiornamenti, invece, possono specificare solo i valori per le variabili appartenenti al modulo. In generale, quindi, un modulo può leggere le variabili di qualsiasi altro modulo, ma scrivere solo le variabili locali, interne al modulo.

Nella sua forma generale un comando è espresso nel seguente modo:

$$[]\mathit{guardia} \rightarrow \mathit{aggiornamento/i}$$

Un esempio di comando potrebbe essere:

$$[]x = 0 \rightarrow 0.8 : (x' = 0) + 0.2 : (x' = 1);$$

$x = 0$ è la guardia del comando, e specifica cosa succede ogni qualvolta x diventa uguale a zero, ovvero descrive il comportamento del modulo quando è verificata la condizione di guardia.

$(x' = 0)$ e $(x' = 1)$ sono gli aggiornamenti a cui è stata assegnata una certa probabilità.

Il significato di questo comando è: ogni qualvolta la variabile x assume il valore zero può rimanere sempre in zero ($x' = 0$) con probabilità 0.8 oppure passare allo stato uno ($x' = 1$) con probabilità 0.2.

Variabili Globali

In aggiunta alle variabili locali appartenenti a ciascun modulo, un modello PRISM può anche includere variabili globali, che possono essere scritte e/o lette da tutti i moduli. Come le variabili locali, queste possono assumere valori interi o booleani. Le variabili globali sono dichiarate in modo identico alle variabili locali di un modulo, salvo che la dichiarazione deve essere esterna alla definizione di qualsiasi modulo.

Sincronizzazione

Prism permette di etichettare i comandi con le azioni (label). Queste label sono posizionate all'interno delle parentesi quadre che segnano l'inizio del comando.

Esempio:

$$[label]comando$$

Queste azioni possono essere utilizzate per forzare due o più moduli a transitare contemporaneamente (cioè a sincronizzarsi), ovvero i comandi dei moduli che avranno la stessa label verranno eseguiti tutti nello stesso istante.

3.1.2 Specifica delle Proprietà

Per analizzare un modello probabilistico che è stato specificato e costruito in PRISM, è necessario identificare una o più proprietà del modello che possono essere analizzate dal tool.

Uno dei compiti fondamentali, quando si specificano le proprietà di un modello, è quello di identificare particolari gruppi o classi di stati del modello.

Ad esempio, per verificare una proprietà come "l'algoritmo alla fine termina con successo con probabilità 1", è prima necessario identificare gli stati del modello che corrispondono a situazioni in cui "l'algoritmo ha terminato con successo". In PRISM, ciò si ottiene semplicemente scrivendo un'espressione che restituisce un valore booleano. Questa espressione in genere deve contenere riferimenti a variabili (e costanti) del modello a cui essa si riferisce. L'insieme degli stati corrispondenti a questa espressione è quello per cui l'espressione restituisce true, ovvero il sistema ci dice che l'espressione è "soddisfatta" in questi stati.

L'Operatore P

Uno dei più importanti operatori del linguaggio di specifica di PRISM è l'operatore P, che viene utilizzato per ragionare sulla probabilità del verificarsi di un evento. L'operatore P è applicabile a tutti i tipi di modelli supportati da PRISM.

Un esempio della semantica dell'operatore **P** potrebbe essere:

$$\mathbf{P}\ bound\ [pathprop]$$

Questa proprietà è vera in uno stato s di un modello, se "la probabilità che la proprietà di percorso pathprop è soddisfatta dai percorsi raggiunge il limite bound". Un tipico esempio di un limite potrebbe essere maggiore di un certo valore, ovvero:

$$\mathbf{P} > 0.98[pathprop]$$

Proprietà Quantitative

Molto spesso è utile adottare un approccio quantitativo per il controllo del modello probabilistico, ovvero calcolare la probabilità effettiva che alcuni compor-

tamenti siano osservati dal modello, piuttosto che verificare se la probabilità è al di sopra o al di sotto di una determinata soglia.

L'operatore P, in questo caso, prende la forma:

$$\mathbf{P} \rightarrow [pathprop]$$

A differenza del caso precedente, questa proprietà ritorna un valore numerico.

Proprietà di Percorso

Una proprietà di percorso è una formula che restituisce true oppure false per un singolo percorso in un modello. I tipi fondamentali di proprietà di percorso che possono essere utilizzati all'interno dell'operatore P sono:

- **X** : "next"
- **U** : "until"
- **F** : "eventually" oppure "future"
- **G** : "always" oppure "globally"
- **W** : "weak until"
- **R** : "relase"

Next

La proprietà **X** prop è vera se prop è vera nel suo secondo stato, ovvero nello stato successivo.

Until

La proprietà prop1 **U** prop2 è vera per un percorso se prop2 è vera in uno stato del percorso e prop1 è vera in tutti gli stati precedenti.

Eventually

La proprietà **F** prop è vera per un percorso se prop alla fine diventa vera in qualche punto lungo il percorso. L'operatore F è un caso speciale dell'operatore U, infatti spesso possiamo trovare F prop scritta come true U prop.

Globally

A differenza dell'operatore F che viene utilizzato per verificare la raggiungibilità, l'operatore **G** viene utilizzato per verificare l'invarianza. Infatti la proprietà G prop è vera per un percorso se prop rimane vera in tutti gli stati lungo il percorso.

Come F e G, gli operatori **W** e **R** sono derivabili da altri operatori temporali.

Capitolo 4

Definizione ed Implementazione della Metodologia

Concentrando l'attenzione su quanto realizzato nel lavoro di tesi, nelle prossime sezioni verranno dettagliate le fasi di creazione del modello e di definizione delle proprietà. Come primo step verranno descritti i dati presi in input dalla metodologia.

4.1 Dati del Modello

La base dati analizzata riguarda:

- file "perturbation"
- file "timeseries_discretized"

Il primo file contiene le perturbazioni applicate alla rete in analisi, ed assume la forma riportata in figura 4.1.

Come possiamo notare, ogni gene può essere attivato (valore 1.0), disattivato (valore -1.0) oppure può essere lasciato libero di evolvere (valore 0.0).

purR	prs	speA	purK
0.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0
-1.0	0.0	0.0	0.0
0.0	1.0	0.0	0.0
0.0	-1.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.0	-1.0	0.0
0.0	0.0	0.0	1.0
0.0	0.0	0.0	-1.0

Figura 4.1: Perturbazioni per una rete di 4 geni

Questo file rispecchia le perturbazioni applicate durante la tecnica microarray. Possiamo notare come, in una generica rete con n geni, possiamo avere 3^n perturbazioni possibili, nella realtà le perturbazioni che possono essere applicate sono molte di meno, e di conseguenza il sistema deve essere in grado di fornire risultati accettabili anche con un numero molto ridotto di perturbazioni a disposizione.

Il secondo file timeseries_discretized assume, invece, la forma riportata in figura 4.2.

Ogni perturbazione presente nel file perturbations corrisponde ad un intero blocco nel file timeseries, ed ogni blocco del file timeseries rappresenta l'evoluzione della rete analizzata in risposta alla perturbazione applicata.

Le timeseries ci permettono di rappresentare l'evoluzione temporale dei geni che appartengono alla rete di interesse. In particolare, ogni riga corrisponde ad un determinato timeslot e descrive lo stato dei geni in un particolare istante di tempo. Pertanto l'insieme di tutti i timeslot riassume l'evoluzione dello stato dei geni in ogni istante di tempo.

L'algoritmo prende in considerazione l'evoluzione temporale dell'espressione genica in modo discretizzato. La discretizzazione è stata effettuata in accordo a quanto riportato in [18, 13]. La tecnica di discretizzazione impie-

"Time"	purR	prs	speA	purK
0	+1	+1	+1	+1
1	+1	+1	+1	+1
2	+1	+1	+1	+1
3	+1	+1	+1	+1
4	+1	+1	+1	0
5	+1	0	0	0
6	+1	0	0	−1
7	+1	−1	−1	−1
8	+1	−1	−1	−1
9	+1	−1	−1	−1
10	+1	−1	−1	−1
11	+1	−1	−1	−1
12	+1	−1	−1	−1
13	+1	−1	−1	−1
14	+1	−1	−1	−1
15	+1	−1	−1	−1
16	+1	−1	−1	−1
17	+1	−1	−1	−1
18	+1	−1	−1	−1
19	+1	−1	−1	−1
20	+1	−1	−1	−1

Figura 4.2: Timeseries corrispondente alla prima purturbazione (presente in 4.1). Il file completo contiene tante timeseries quante sono le perturbazioni.

gata divide i valori dell'espressione genica in tre livelli, ognuno contenente approssimativamente lo stesso numero di valori di espressione.

Il nostro intento sarà quello di ricavare una rete quanto più accurata possibile, partendo da questi dati in analisi, applicando tecniche di Reverse Engineering.

Infine, come output del sistema viene prodotto un file contenente la rete genica ottenuta, a valle dell'applicazione dell'algoritmo.

4.2 Il Modello

Il modello prodotto nel nostro lavoro simula quello che è il comportamento dei geni nelle varie TimeSeries a nostra disposizione.

```
module PERTURBATION_CHOICE

        c_state_p : [0..2];
        a_state_p : [0..2];
        b_state_p : [0..2];
        count : [0..1] init 0;

        // stato iniziale in cui viene SCELTA LA PERTURBAZIONE
        [] perturb_num=n -> 1/n:(perturb_num'=0) + 1/n:(perturb_num'=1) + 1/n:(perturb_num'=2) + 1/n:(perturb_num'=3) + 1/n:(perturb_num'=4) +
                            1/n:(perturb_num'=5) + 1/n:(perturb_num'=6) + 1/n:(perturb_num'=7) + 1/n:(perturb_num'=8) + 1/n:(perturb_num'=9) +
                            1/n:(perturb_num'=10) +
                                ......


        [st0] perturb_num=0 & count=0 -> (c_state_p'=1) & (a_state_p'=1) & (b_state_p'=1) & (count'=1);
        [st0] perturb_num=1 & count=0 -> (c_state_p'=1) & (a_state_p'=1) & (b_state_p'=2) & (count'=1);
        [st0] perturb_num=2 & count=0 -> (c_state_p'=1) & (a_state_p'=1) & (b_state_p'=0) & (count'=1);
        [st0] perturb_num=3 & count=0 -> (c_state_p'=1) & (a_state_p'=2) & (b_state_p'=1) & (count'=1);
        [st0] perturb_num=4 & count=0 -> (c_state_p'=1) & (a_state_p'=2) & (b_state_p'=2) & (count'=1);
        [st0] perturb_num=5 & count=0 -> (c_state_p'=1) & (a_state_p'=2) & (b_state_p'=0) & (count'=1);
        [st0] perturb_num=6 & count=0 -> (c_state_p'=1) & (a_state_p'=0) & (b_state_p'=1) & (count'=1);
        [st0] perturb_num=7 & count=0 -> (c_state_p'=1) & (a_state_p'=0) & (b_state_p'=2) & (count'=1);
        [st0] perturb_num=8 & count=0 -> (c_state_p'=1) & (a_state_p'=0) & (b_state_p'=0) & (count'=1);
        [st0] perturb_num=9 & count=0 -> (c_state_p'=2) & (a_state_p'=1) & (b_state_p'=1) & (count'=1);
        [st0] perturb_num=10 & count=0 -> (c_state_p'=2) & (a_state_p'=1) & (b_state_p'=2) & (count'=1);
        ......

endmodule
```

Figura 4.3: Frammento del modulo che descrive la scelta della perturbazione

In accordo con la sintassi del linguaggio Prism sono stati sviluppati vari moduli che eseguono in parallelo. I due tipi di moduli prodotti possono essere così riassunti: un modulo che simula il comportamento della scelta della perturbazione (PERTURBATION_CHOISE) e un modulo (GENE_NOMEGENE) per ogni gene, che simula il comportamento del gene relativo ad ogni perturbazione.

Quest'ultimo tipo di modulo segue l'andamento del gene, ovvero il variare del suo valore, attraverso i vari stati della Timeserie.

Le figure 4.3 e 4.4 riportano gli esempi dei due tipi di moduli creati.

Funzionamento del Modello

Nel nostro modello il primo passo che viene effettuato è quello della scelta della perturbazione. Le perturbazioni sono tutte equiprobabili e hanno probabilità pari a $1/n$ (con n uguale al numero di perturbazioni a disposizione). Una volta scelta la perturbazione si passa all'esecuzione della timeserie corrispondente. I valori assunti dai geni nei vari timeslots sono stati sincronizzati utilizzando delle label.

```
module GENE_C

        c_state : [0..3] init 3;
        cC : [0..51] init 0;

        // PERTURBAZIONE -> 0
        // stato iniziale PERTURBAZIONE 0
        [st0] c_state=3 & perturb_num=0 -> (c_state'=2) & (cC'=1);

        [st1] c_state=2 & cC=1 & perturb_num=0 -> (c_state'=2) & (cC'=2);
        [st2] c_state=2 & cC=2 & perturb_num=0 -> (c_state'=2) & (cC'=3);
        [st3] c_state=2 & cC=3 & perturb_num=0 -> (c_state'=2) & (cC'=4);
        ...

        // PERTURBAZIONE -> 1
        // stato iniziale PERTURBAZIONE 1
        [st0] c_state=3 & perturb_num=1 -> (c_state'=2) & (cC'=1);

        [st1] c_state=2 & cC=1 & perturb_num=1 -> (c_state'=2) & (cC'=2);
        [st2] c_state=2 & cC=2 & perturb_num=1 -> (c_state'=2) & (cC'=3);
        [st3] c_state=2 & cC=3 & perturb_num=1 -> (c_state'=2) & (cC'=4);
        [st4] c_state=2 & cC=4 & perturb_num=1 -> (c_state'=2) & (cC'=5);
        ...

        ......

        // PERTURBAZIONE -> n
        ...

endmodule
```

Figura 4.4: Frammento del modulo che descrive l'espressione del gene

4.3 Proprietà

Come già affermato in precedenza, per analizzare un modello probabilistico, specificato e costruito in PRISM, è necessario identificare una o più proprietà del modello che possono essere analizzate dal tool stesso.

Per definire le proprietà da verificare è stata fatta un'analisi manuale a priori dei dati a disposizione. Per facilitare l'analisi sono state utilizzate reti di soli 3 geni di cui si conosceva la rete di regolazione genica reale.

4.3.1 Analisi Preliminare dei Dati

Partendo dagli archi risultanti sono stati osservati i valori assuunti dai geni, al variare delle perturbazioni, nelle timeseries corrispondenti. Così facendo si è potuta osservare la corrispondenza esistente tra gli archi e i valori assunti dai geni al variare delle perturbazioni.

Risultati Osservati

In presenza di reti di soli archi attivatori si è visto che i geni, indipendentemente da se essi venivano attivati o erano essi stessi attivatori, solitamente raggiungevano il valore di up (valore logico alto) e rimanevano up per tutti gli istanti di tempo considerati nella timeserie. In caso di presenza di archi inibitori questo non succedeva, ovvero il valore del gene, anche se diventava up, il suo valore non rimaneva up. Da questa analisi è stata prodotta la seguente proprietà che verifica appunto se un gene diventa up e resta up:

$$P =?[F(c\_state = 2 \,\&(G\ c\_state = 2^1))]$$

Naturalmente questa proprietà non è verificata quando il gene viene inibito nella perturbazione.

E' stato inoltre osservato che, in alcuni casi, dei geni assumevano valore logico alto durante tutta l'evoluzione temporale, pertanto si è deciso di verificare la sottostante proprietà:

$$P =?[c\_state = 3\ \ U\ (c\_state = 2\ \&\ (G\ c\_state = 2))]$$

Infine si è deciso di verificare se qualche gene parte sempre a valore up:

$$P =?[c\_state = 3\ \ U(c\_state = 2\ \&\ (G\ c\_state = 2))]$$

Le proprietà sopra menzionate sono scritte nella sintassi di prism e sono tutte relative al generico gene C; naturalmente queste proprietà sono state scritte e verificati per ogni gene appartenente alla rete.

[1] Il 2 rappresenta il valore logico alto. PRISM non permette l'utilizzo di numeri negativi, quindi gli stati $(-1, 0, 1)$ sono stati rispettivamente rappresentati con $(0, 1, 2)$.

	1	2	3
A	0	0	1
B	0	0	1
C	1	1	1

Tabella 4.1: Matrice Risultati: il valore 1 corrisponde a TRUE e il valore 0 a FALSE.

Alcune Considerazioni

L'unione di queste 3 proprietà e l'analisi dei risultati che generano porta all'individuazione dei geni forti[2] e dei geni inibiti.

Per osservare meglio i risultati ottenuti si è preferito organizzarli in una matrice $n \times m$, con la cardinalità di riga pari al numero di geni e la cardinalità di colonna pari a 3, ovvero una colonna per ogni controllo. Nella tabella 4.1 è riportato un esempio di matrice per una rete di 3 geni.

Andando ad analizzare i risultati contenuti nella matrice è possibile fare delle considerazioni preliminari sulla rete, ovvero:

- se nella rete sono presenti geni forti;
- se nella rete ci sono sia archi attivatori che archi inibitori, quindi la presenza di geni inibiti;
- se la rete è di soli archi attivatori oppure di soli archi inibitori;

Per individuare la presenza di geni forti viene effettuato un controllo per riga nella matrice, ovvero per ogni gene viene verificato se tutti i controlli hanno dato come risultato true. Un esempio di gene che verifica questo controllo è riportato in tabella 4.2.

[2]Per gene forte si intende un gene che assume sempre il valore logico alto tranne quando viene inibito.

	1	2	3
A	0	0	1
B	0	0	1
C	**1**	**1**	**1**

Tabella 4.2: Il gene C è un gene Forte.

La presenza di geni forti all'interno della rete ci permette di fare delle semplificazioni nella costruzione della rete finale, ovvero partendo da una rete completa possiamo eliminare, dall'insieme degli archi attivatori, tutti gli archi che hanno il gene forte a destra della relazione che individua l'arco (ad esempio: se C è un gene forte in una rete di solo 3 geni, gli archi $A \rightarrow C$ e $B \rightarrow C$ vanno rimossi).

Questa considerazione trova fondamento poiché, partendo dal presupposto che la relazione che individua un arco attivatore è: l'attivazione di un gene porta un altro gene ad attivarsi, essa non deve essere vera per geni che sono sempre attivi, poiché su questi geni non è possibile stabilire una relazione causa-effetto. Infatti, valutando questa proprietà, per archi che hanno geni forti a destra del simbolo di relazione, i risultati sono positivi. Questi risultati però sono ovvi, per geni sempre attivi, e naturalmente devono essere esclusi.

I geni inibiti, invece, vengono individuati andando a controllare i risultati relativi alla prima proprietà sopra enunciata, ovvero andando a controllare nella matrice la colonna corrispondente alla proprietà. La matrice, riportata in tabella 4.3, ne è un esempio.

Se la proprietà risultasse essere verificata per ogni gene presente nella rete (ovvero la colonna della proprietà assume valore true per ogni gene) allora vuol dire che siamo in presenza di una rete di soli archi attivatori, di conseguenza nella verifica successiva che controlla quali archi sono presenti nella rete

	1	2	3
A	0	0	**0**
B	0	0	1
C	1	1	1

Tabella 4.3: La terza colonna della matrice corrisponde alla prima proprietà. In questo esempio il gene A non verifica il controllo.

possiamo omettere di controllare gli archi inibitori.

Se il controllo, invece, non dovesse dare risultato positivo (ovvero nella colonna della proprietà qualche gene ha valore falso) allora vuol dire che ci troviamo in presenza di geni inibiti. In questo caso oltre a verificare le proprietà sugli archi attivatori vanno verificate anche le proprietà relative agli archi inibitori.

Infine possiamo verificare se la rete in esame è una rete di soli archi inibitori. Per verificare ciò si va a controllare se la somma tra il numero di geni forti e il numero di geni inibiti è pari al numero di geni presenti nella rete e se il numero di geni forti è maggiore di zero. Nel caso in cui questo controllo dovesse dare esito positivo allora vorrebbe dire che ci troviamo di fronte ad una rete di soli archi inibitori, e in questo caso non occorre verificare le proprietà che individuano gli archi attivatori.

Per riassumere, il controllo che si effettua è il seguente:

$$((\sharp geniForti + \sharp geniInibiti) == \sharp geniTotali) \;\&\&\; (\sharp geniForti > 0) \quad (4.1)$$

Ricapitolando quindi, l'analisi dei risultati prodotti dalla tre proprietà sopra enunciate ci ha permesso di fare considerazioni preliminari nella costruzione della rete finale:

- Se nessun gene viene inibito ci troviamo in presenza di una rete di soli archi attivatori, quindi nell'individuazione degli archi reali possiamo

applicare solo le proprietà che individuano gli archi attivatori.

- Se un gene risulta essere un gene forte, vanno rimossi dalla rete tutti gli archi attivatori che hanno il gene forte a destra nella relazione che individua l'arco.

- Se dovesse essere verificata la seguente relazione (4.1): la rete in esame è una rete di soli archi inibitori quindi nell'individuazione degli archi reali possiamo applicare solo le proprietà che individuano gli archi inibitori.

4.3.2 Proprietà che Individuano gli Archi della Rete

Come già accennato in precedenza ci sono altre proprietà che vengono verificate sul modello, e queste sono quelle relative agli archi attivatori e agli archi inibitori.

Queste proprietà sono state definite nel seguente modo:

- per individuare se un arco è attivatore: dati due geni X e Y, tra i due geni c'è un arco di tipo attivatore, ovvero $X \rightarrow Y$, se quando X diventa alto anche Y diventa alto e dopo non ritorna più basso;

- per individuare se un arco è inibitore: dati due geni X e Y, tra i due geni c'è un arco di tipo inibitore, ovvero $X \dashv Y$, se quando X diventa alto Y diventa basso e dopo non ritorna più alto.

Le proprietà sopra elencate sono proprietà assai stringenti poiché c'è una condizione di permanenza di stato sul secondo gene. Purtroppo dall'analisi preliminare fatta sui dati è emerso che in presenza di geni inibiti, e quindi di una rete che ha anche archi inibitori, una relazione così stringente non considera tutti i casi possibili e quindi porta a dei risultati errati. Questo accade perché in presenza di archi inibitori non è detto che, ad esempio, un gene diventi alto

e dopo non ritorni più basso; esso può diventare basso proprio a causa di un altro gene che lo potrebbe inibire.

Considerato questo si è dunque deciso di introdurre altre due proprietà, dette proprietà weak, che sono più deboli delle precedenti:

- per individuare se un arco è attivatore: dati due geni X e Y, tra i due geni c'è un arco di tipo attivatore, ovvero $X \rightarrow Y$, se quando X diventa alto anche Y diventa alto;
- per individuare se un arco è inibitore: dati due geni X e Y, tra i due geni c'è un arco di tipo inibitore, ovvero $X \dashv Y$, se quando X diventa alto Y diventa basso;
- come si può notare la forma della proprietà è rimasta la stessa, l'unica cosa che è cambiata è la condizione di permanenza, ovvero si considerano anche i casi in cui rispettivamente il gene ritorna basso e il gene ritorna alto.

Queste ultime proprietà elencate sono riportate di seguito:
Per individuare gli archi attivatori, la proprietà è la seguente:

$$P =?[(c\_state <= 3 \ U \ (c\_state = 2)) \ U$$

$$(c\_state = 2 \ \& \ (F(a\_state = 2 \ \& \ (Ga\_state = 2))))]^3$$

questa invece è la sua versione WEAK:

$$P =?[(c\_state <= 3 \ U \ (c\_state = 2)) \ U \ (c\_state = 2 \ \& \ (F(a\_state = 2)))]$$

Per individuare gli archi inibitori, la proprietà è la seguente:

$$P =?[(c\_state <= 3 \ U \ (c\_state = 2)) \ U$$

[3]La formula individua l'arco $C \rightarrow A$

$$(c\_state = 2 \ \& \ (F(a\_state = 0 \ \& \ (Ga\_state = 0))))]$$

questa invece è la sua versione WEAK:

$$P =?[(c\_state <= 3 \ \ U \ \ (c\_state = 2)) \ \ U \ \ (c\_state = 2 \ \& \ (F(a\_state = 0)))]$$

Per rafforzare ulteriormente i controlli, indipendentemente dal tipo di rete, si è deciso di introdurre altre quattro proprietà, che altro non sono che il complemento di quelle appena descritte. Si è deciso, quindi di controllare le seguenti proprietà:

Arco attivatore:

$$P =?[(c\_state <= 3 \ \ U \ \ (c\_state = 0)) \ \ U$$

$$(c\_state = 0 \ \& \ (F(a\_state < 2 \ \& \ (Ga\_state < 2))))]$$

Arco inibitore:

$$P =?[(c\_state <= 3 \ \ U \ \ (c\_state = 0)) \ \ U$$

$$(c\_state = 0 \ \& \ (F(a\_state > 0 \ \& \ (Ga\_state > 0))))]$$

Queste proprietà verificano rispettivamente:

- se un arco è attivatore: dati due geni X e Y, tra i due geni c'è un arco di tipo attivatore, ovvero $X \rightarrow Y$, se quando X diventa basso anche Y diventa basso e non ritorna più alto;
- e un arco è inibitore: dati due geni X e Y, tra i due geni c'è un arco di tipo inibitore, ovvero $X - |Y$, se quando X diventa basso Y diventa alto e non ritorna più basso.

Anche di queste ultime è stata prodotta una versione WEAK:

WEAK arco attivatore:

$$P =?[(c\_state <= 3\ \ U\ \ (c\_state = 0))\ \ U$$

$$(c\_state = 0\ \&\ (F(a\_state < 2)))]$$

WEAK arco inibitore:

$$P =?[(c\_state <= 3\ \ U\ \ (c\_state = 0))\ \ U$$

$$(c\_state = 0\ \&\ (F(a\_state > 0)))]$$

Riassumendo quindi sono state prodotte in tutto otto proprietà. Per facilitarne la comprensione e il riferimento si è deciso di dividerle in due gruppi: le prime 4 proprietà elencate sono di tipo OVEREXPRESSION mentre le ultime 4 elencate sono di tipo KNOCKOUT.

Si è deciso di rinominarle così per via del valore che assume il gene alla sinistra del simbolo di relazione dell'arco; infatti nelle prime 4 proprietà il gene alla sinistra è sempre posto al valore alto, mentre nelle ultime 4 è sempre posto al valore basso. I termini Overexpression e Knockout sono stati scelti quindi in relazione all'espressione del gene che compare alla sinistra nella relazione che indica l'arco.

4.4 Risultati Ottenuti

I risultati che si ottengono dalla verifica delle proprietà di overexpression e di knockout sul modello assegnano ad ogni arco una probabilità, quindi la rete risultante sarà una rete completa in cui ad ogni arco è assegnata una certa probabilità. Per ridurre l'insieme degli archi risultanti si procede eliminandone alcuni in accordo a quanto precedentemente affermato.

Purtroppo questa analisi non ci permette di ottenere sempre gli archi reali della rete. In altri termini ci si aspetterebbe che gli archi reali siano solo quelli con la probabilità maggiore. In realtà questo non è sempre vero in quanto, in alcuni casi, la probabilità degli archi effettivamente presenti nella rete reale è inferiore rispetto a quella assegnata ad archi che non fanno parte della rete reale.

Dal momento che questa prima analisi non porta sempre a risultati veritieri sono state introdotte altre proprietà che permettono di migliorare l'accuratezza dell'individuazione degli archi reali della rete. Nella prossima sezione verranno descritte queste nuove proprietà.

4.5 Proprietà Finali

Per ovviare al problema di cui sopra si è pensato di introdurre altre proprietà che rispetto alle proprietà già menzionate prendono in considerazione anche la perturbazione. La proprietà, quindi, non è stata applicata a tutte le perturbazioni (e quindi a tutte le timeseries a disposizione) ma viene applicata solo alle perturbazioni (quindi alle rispettive timeseries) in cui il gene alla sinistra del simbolo di relazione dell'arco è bloccato ad un determinato valore. Nello specifico nelle proprietà di overexpression vengono considerate solo le perturbazioni in cui il gene alla sinistra è al valore logico alto, mentre in quelle di knockout vengono considerate solo le perturbazioni in cui il gene alla sinistra è al valore logico basso. Di seguito sono riportate le nuove proprietà. Osservandole si nota subito che sono identiche alle precedenti fatta eccezione solo di questa variabile:

$$c\_state\_p = 2 \ (oppure \ c\_state\_p = 0)$$

che indica appunto il valore di espressione del gene all'interno della perturbazione.

Proprietà per individuare gli archi attivatori

Proprietà che verifica gli archi attivatori OVEREXPRESSION:

$$P =?[(c\_state <= 3\ \ U\ \ (c\_state = 2\ \&\ c\_state\_p = 2))\ \ U$$

$$(c\_state = 2\ \&\ (F(a\_state = 2\ \&\ (Ga\_state = 2))))]$$

Proprietà che verifica gli archi attivatori OVEREXPRESSION WEAK:

$$P =?[(c\_state <= 3\ \ U\ \ (c\_state = 2\ \&\ c\_state\_p = 2))\ \ U$$

$$(c\_state = 2\ \&\ (F(a\_state = 2)))]$$

Proprietà che verifica gli archi attivatori KNOCKOUT:

$$P =?[(c\_state <= 3\ \ U\ \ (c\_state = 0\ \&\ c\_state\_p = 0))\ \ U$$

$$(c\_state = 0\ \&\ (F(a\_state < 2\ \&\ (Ga\_state < 2))))]$$

Proprietà che verifica gli archi attivatori KNOCKOUT WEAK:

$$P =?[(c\_state <= 3\ \ U\ \ (c\_state = 0\ \&\ c\_state\_p = 0))\ \ U$$

$$(c\_state = 0\ \&\ (F(a\_state < 2)))]$$

Proprietà per individuare gli archi inibitori

Proprietà che verifica gli archi inibitori OVEREXPRESSION:

$$P =?[(c\_state <= 3\ \ U\ \ (c\_state = 2\ \&\ c\_state\_p = 2))\ \ U$$

$$(c\_state = 2\ \&\ (F(a\_state = 0\ \&\ (Ga\_state = 0))))]$$

Proprietà che verifica gli archi inibitori OVEREXPRESSION WEAK:

$$P =?[(c\_state <= 3\ \ U\ \ (c\_state = 2\ \&\ c\_state\_p = 2))\ \ U$$

$$(c\_state = 2 \ \& \ (F(a\_state = 0)))]$$

Proprietà che verifica gli archi inibitori KNOCKOUT:

$$P =?[(c\_state <= 3 \ \ U \ \ (c\_state = 0 \ \& \ c\_state\_p = 0)) \ \ U$$

$$(c\_state = 0 \ \& \ (F(a\_state > 0 \ \& \ (Ga\_state > 0))))]$$

Proprietà che verifica gli archi inibitori KNOCKOUT WEAK:

$$P =?[(c\_state <= 3 \ \ U \ \ (c\_state = 0 \ \& \ c\_state\_p = 0)) \ \ U$$

$$(c\_state = 0 \ \& \ (F(a\_state > 0)))]$$

4.6 Algoritmo Risolutivo

A partire dai risultati ottenuti dall'applicazione delle precedenti proprietà è stato ideato un algoritmo che prende in ingresso i risultati delle varie proprietà e attraverso vari passi costruisce la rete finale risultante. L'algoritmo esegue i seguenti passi:

1. controlla se nella rete sono presenti geni forti, in caso affermativo elimina tutti gli archi che hanno i geni forti alla destra del simbolo che indica la relazione dell'arco

2. eliminana tutti gli archi che hanno valori di overexpression e knockout[4] troppo distanti tra di loro, ovvero la differenza tra il risultato della proprietà overexpression e quello della proprietà di knockout è maggiore di un certo ϵ.

3. eliminana tutti gli archi che hanno dato come risultato, dell'applicazione delle proprietà di overexpression e knockout, almeno un valore di probabilità pari a zero

[4]Le proprietà utilizzate in questo caso sono le forti e non le weak.

4. vengono eliminati tutti gli archi che hanno una differenza tra valore di overexpression e overexpression weak (oppure tra valore di knockout e knockout weak) maggiore di una certa soglia

5. effettua dei raggruppamenti in funzione dei risultati delle proprietà di overexpression* e di knockout* (non weak). Ogni raggruppamento contiene i soli archi che risultano avere la stessa probabilità

6. seleziona i soli raggruppamenti costituiti dal maggior numero di archi, ovvero il massimo raggruppamento

7. infine, nel caso di due o più raggruppamenti massimi, seleziona quello con la probabilità massima

Specifica dell'ϵ

La formula per il calcolo dell'epsilon, utilizzato nel secondo passo dell'algoritmo come soglia per eliminare alcuni archi, è la seguente:

$$\epsilon = \frac{3^{(n-1)}}{\frac{tabelleTotali}{3}}$$

Questa soglia confronta il valore della probabilità ottenuto dall'arco con il numero dei geni totali della rete meno quello bloccato al valore della perturbazione (scelta nella formula) e lo divide per il numero totale di timeseries a disposizione. L'ultima divisione per tre è stata introdotta per differenziare i valori possibili che i geni possono assumere,, ovvero il valore logico basso, il valore logico alto e il valore basale.

Gli archi che hanno ottenuto un valore di probabilità, dato dalla differenza tra il risultato della proprietà overexpression e quello della proprietà di knockout, superiore ad ϵ devono essere eliminati poiché questo significherebbe che le

due proprietà, che sono una il complemento[5] dell'altra, hanno dato dei risultati troppo scostanti tra di loro. Questo significa che con molta probabilità l'arco in questione non esiste. I risultati della differenza tra overexpression e knockout non possono scostarsi molto tra loro, perché come detto in precedenza le due proprietà sono una il complemento dell'altra.

Alcune Considerazioni sull'ϵ

Sulla soglia appena introdotta bisogna fare alcune considerazioni. Questo controllo è rilevante solo per reti che hanno un numero di perturbazioni, e quindi un numero di timeseries, elevato ($\simeq 3^n$, con n pari al numero di geni in ingresso). Se si hanno a diposizione, quindi, un numero di perturbazioni completo questo controllo diventa fondamentale, poiché sfoltisce di molto gli archi presenti nella rete. Se così non fosse, ovvero se si avessero a disposizione meno perturbazioni, questo controllo non ha nessun effetto sugli archi della rete, e tutti andrebbero in ingresso ai passi successivi dell'algoritmo.

Specifica della Soglia

La soglia utilizzata nel quarto passo dell'algoritmo è la seguente:

$$\frac{1}{n}$$

Suddetta soglia serve ad eliminare dei risultati troppo distanti tra loro. Questa soglia va a controllare di quando si discostano i risultati ottenuti dall'applicazione delle proprietà nella versione forte e in quella weak. Se questi

[5]Si rammenta che per complemento si intende il complemento dello stato del gene analizzato dalle proprietà; esse, infatti verificano sempre la presenza dello stesso arco una volta considerando il gene di sinistra attivo e successivamente considerando il gene di sinistra disattivo. Per maggiori chiarimenti consultare la sezione relativa alle proprietà prodotte.

risultati dovessero essere troppo distanti tra loro potrebbe significare che l'arco non è detto che sia presente.

Per migliorare ulteriormente l'algoritmo si potrebbe pensare di modificare questa soglia. Infatti se si considera un valore di soglia più elevato molti più archi potrebbero entrare a far parte dei possibili archi della rete. Questa soglia la si potrebbe, quindi, utilizzare per cercare di individuare gli archi che hanno alla destra dei geni che potrebbero comparire in relazioni opposte. In altre parole, considerando un arco attivatore che ha alla destra del simbolo di relazione un gene che compare anche alla destra di un arco inibitore. L'individuazione di quest'ultimi archi è molto difficile.

Capitolo 5

Analisi Sperimentale

5.1 Misurazione delle Performance

Le prestazioni degli algoritmi di Reverse Engineering per l'inferenza delle Reti di Regolazione Genica vengo confrontate in termini di capacità di ricostruzione della vera e propria rete di regolazione. Due misure largamente utilizzate, prese in prestito dalla comunità che si occupa di Information Retrieval, sono la *Precision* (P) e la *Recall* (R) dove la *Recall* è la frazione delle connessioni vere che un sistema è capace di predire e la *Precision* è la frazione delle predizioni corrette. Le due misure sono definite come:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

dove TP è il numero dei true positive, ossia il numero di relazioni causali correttamente identificate dall'algoritmo, FP è il numero dei false positive, vale a dire il numero di relazioni individuate dall'algoritmo che non sono corrette, infine FN è il numero dei false negative, cioè il numero di relazioni

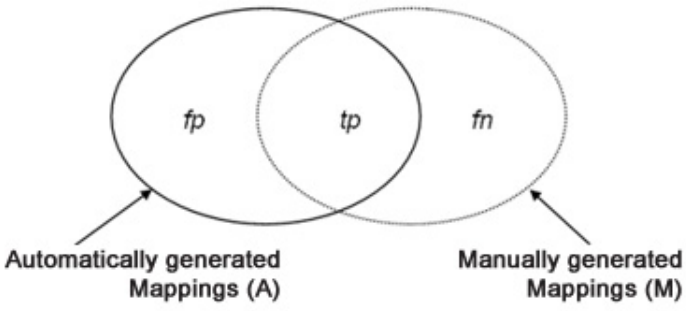

Figura 5.1: Misure di rilevanza: Precision e Recall.

presenti nella rete reale ma non identificate. Un riscontro grafico per illustrare ulteriormente le due misure è dato in figura 5.1.

In un processo di classificazione, i termini vero positivo (TP), vero negativo (TN), falso positivo (FP) e falso negativo (FN) sono usati per confrontare la classificazione di un oggetto (l'etichetta di classe assegnata all'oggetto da un classificatore) con la corretta classificazione desiderata (la classe a cui in realtà appartiene l'oggetto). Entrambe le misure variano nell'intervallo $[0, 1]$.

Un'altra misura di valutazione è la F-measure. Questo parametro è un indice di bontà dell'algoritmo, ed è definito come una combinazione lineare dei due parametri precedenti:

$$F_\beta = \frac{1}{\alpha\frac{1}{p} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

con

$$\beta^2 = \frac{1-\alpha}{\alpha}$$

per $\alpha[0, 1]$, e quindi $\beta^2 \in [0, \infty]$. Valori di $\beta < 1$ danno maggior peso alla *Precision*, mentre valori di $\beta > 1$ danno maggior peso alla *Recall*. La misura equilibrata di riferimento della F-Measure, che pesa ugualmente *Precision* e

Recall, è comunemente nota come $F1$ (abbreviazione $F\beta = 1$), poiché pone $\beta = 1$ (ossia $\alpha = 1/2$). In questo caso la formula si semplifica in:

$$F1 = \frac{2PR}{P + R}$$

La F-Measure può variare fra 0 e 1, ed 1 è il caso ottimale in cui sia la *Precision* che la *Recall* sono pari a 1.

5.2 Risultati Ottenuti

I risultati ottenuti dall'algoritmo sono stati valutati in termini di Precision, Recall ed F-measure. In particolare, va necessariamente adattato il significato attribuito ai 3 (veri positivi, falsi positivi, falsi negativi) parametri utilizzati per il calcolo delle tre metriche di confronto:

- *veri positivi (VP)*: archi individuati dall'algoritmo effettivamente presenti nella rete reale;
- *falsi positivi (FP)*: archi individuati dall'algoritmo che non figurano nella rete reale;
- *falsi negativi (FN)*: archi non individuati dall'algoritmo ma presenti nella rete reale.

I test sono stati effettuati su reti aventi un numero crescente di geni, in particolare si va da un numero minimo di 4 geni ad un numero massimo di 20 geni. Fissato il numero dei geni, i test sono stati eseguiti considerando dieci reti differenti. Per ciascun test abbiamo calcolato i parametri di cui sopra i quali sono dettagliati nelle sezioni seguenti.

Reti 4 geni

Il primo test preso in considerazione valuta 10 reti ognuna contenente 4 geni. Come mostrato nella tabella sottostante (5.2), possiamo immediatamente osservare che la Precision è sempre pari ad 1, pertanto il nostro algoritmo non individua mai dei falsi positivi; la metodologia applicata risulta accurata, ovvero gli archi individuati sono effettivamente tutti presenti nella rete reale. Purtoppo, la Recall non assume sempre valore pari ad 1, il che significa che l'algoritmo non individua qualche arco che effettivamente è presente. Quanto detto ci fa affermare che la metodologia non ci permette di garantire la completezza. Essendo, però, la Recall media pari a 0.86, un valore prossimo all'unità, in definitva l'informazione persa risulta essere minima.

Per valutare la bontá dell'algoritmo prendiamo in considerazione l ' F - measure, in particolare il suo valore medio, che, nel caso in esame, risulta pari a 0.9. Tale valore ci fa dire che, per reti di 4 geni, l'algoritmo raggiunge un livello di accuratezza pari al 90%.

RETI 4 GENI						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	2	0	2	1,000	0,500	0,667
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	2	0	2	1,000	0,500	0,667
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	3	0	2	1,000	0,600	0,750

Figura 5.2: Risultati ottenuti su una rete di 4 geni.

Reti 5 geni

Successivamente abbiamo valutato la bontà della metodologia proposta considerando 10 reti ognuna contenente 5 geni. Valutando i risultati ottenuti,

riporati nella tabella sottostante (5.3), possiamo vedere un lieve calo prestazionale rispetto al caso analizzato in precedenza. In particolare, la Precision in due casi su dieci mostra un valore inferiore all'unità il che implica la presenza di falsi positivi. Ciò nonostante la Precision media risulta pari a 0.96 pertanto il calo prestazionale risulta essere sufficientemente lieve, quasi impercettibile. Quanto detto in termini di Precision si ripercuote anche nel calcolo delle altre due metriche: Recall ed F-measure. Ancora una volta, però i risultati medi sono superiori a 0.85 pertanto l'accuratezza della metodologia resta sufficientemente elevata ($> 85\%$).

RETI 5 GENI						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	4	1	1	0,800	0,800	0,800
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	3	0	1	1,000	0,750	0,857
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	4	1	1	0,800	0,800	0,800
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	3	0	2	1,000	0,600	0,750
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	3	0	1	1,000	0,750	0,857

Figura 5.3: Risultati ottenuti su una rete di 5 geni.

Reti 6 geni

Un ragionamento analogo è stato effettuato considerando 10 reti ognuna contenente 6 geni. In questo caso, analogamente al caso delle reti di 4 geni, la Precision assume sempre valore pari ad 1. Purtroppo abbiamo un lieve calo prestazionale in termini di Recall infatti la Recall media misurata assume valore pari ad 0.82. Questo impatta necessariamente in modo negativo il valore della F-measure che, nel caso in esame, scende al di sotto di 0.9 (F-measure media pari a 0.88). In definitiva possiamo comunque affermare che l'algoritmo è in grado individuare sempre archi presenti nella rete anche se perde più in-

formazione rispetto ai casi analizzati in precedenza. Ciò che viene impattato è quindi la completezza della metodologia. I risultati sono riportati nella tabella seguente (5.4).

RETI 6 GENI						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	4	0	1	1,000	0,800	0,889
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	4	0	2	1,000	0,667	0,800
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	3	0	4	1,000	0,429	0,600
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	5	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	5	0	2	1,000	0,714	0,833
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	5	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	5	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	7	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	3	0	2	1,000	0,600	0,750
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	5	0	0	1,000	1,000	1,000

Figura 5.4: Risultati ottenuti su una rete di 6 geni.

Reti 10 geni

Relativamente alle reti contenenti 10 geni possiamo affermare che i risultati delle tre metriche assumono sempre valori medi inferiori all'unità. Inoltre, i valori ottenuti sono inferiori a quelli analizzati in precedenza. Questo ci permette di affermare che un incremento del numero dei geni porta inevitabilmente ad un calo delle prestazioni. Nel complesso però l'algoritmo presenta un livello di accuratezza $> 75\%$ e un livello di completezza $\simeq 70\%$. I risultati ottenuti sono riportati nella tabella riportata in figura 5.5.

Reti 15 geni

Un altro caso analizzato prende in considerazione reti contenenti 15 geni. In questo caso, abbiamo un risultato sorprendente: le tre metriche assumono un valore superiore a quello assunto nel caso di 10 geni. Questo risultato lo otteniamo in quanto migliora nettamente la Precision dal momento che solo in tre casi l'algoritmo mostra falsi positivi mentre la Recall assume un valore

RETI 10 GENI						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	8	2	10	0,800	0,444	0,571
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	9	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	5	0	10	1,000	0,333	0,500
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	8	0	1	1,000	0,889	0,941
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	12	2	7	0,857	0,632	0,727
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	8	0	1	1,000	0,889	0,941
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	8	0	1	1,000	0,889	0,941
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	9	0	1	1,000	0,900	0,947
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	9	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	0	16	16	0,000	0,000	0,000

Figura 5.5: Risultati ottenuti su una rete di 10 geni.

pressochè pari a quello misurato in precedenza. L'F-measure media è pari a 0.78 pertanto possiamo dire che la nostra metodologia risulta accurata ($\simeq 80\%$) anche su reti con numero di geni pari a 15.

RETI 15 GENI						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	11	0	3	1,000	0,786	0,880
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	14	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	12	0	2	1,000	0,857	0,923
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	19	0	7	1,000	0,731	0,844
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	12	9	6	0,571	0,667	0,615
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	10	0	6	1,000	0,625	0,769
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	14	2	14	0,875	0,500	0,636
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	8	1	9	0,889	0,471	0,615
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	19	1	14	0,950	0,576	0,717
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	16	0	5	1,000	0,762	0,865

Figura 5.6: Risultati ottenuti su una rete di 15 geni.

Reti 20 geni

L'ultimo caso osservato prende in considerazione reti contenenti 20 geni. Anche in questo caso, come si era verificato nel precedente caso, le tre metriche assumono un valore superiore a quello assunto nel caso di 15 geni. Questo risultato lo otteniamo in quanto migliora ulteriormente la Precision, in soli tre casi l'algoritmo mostra falsi positivi e inoltre migliora anche la Recall, che assume un valore circa uguale a 0.75. L'F-measure media è pari a 0.83 pertanto

possiamo dire che la nostra metodologia risulta accurata (83%) anche su reti con numero di geni pari a 20.

RETI 20 GENI						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	20	0	1	1,000	0,952	0,976
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	17	1	10	0,944	0,630	0,756
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	18	0	1	1,000	0,947	0,973
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	21	0	7	1,000	0,750	0,857
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	17	1	9	0,944	0,654	0,773
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	24	12	33	0,667	0,421	0,516
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	18	0	8	1,000	0,692	0,818
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	17	0	2	1,000	0,895	0,944
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	29	0	12	1,000	0,707	0,829
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	25	0	6	1,000	0,806	0,893

Figura 5.7: Risultati ottenuti su una rete di 20 geni.

5.3 Confronto con FormalM

I risultati contenuti nelle sezionioni precedenti permettono di valutare, in termini di accuratezza e completezza, le prestazioni della metodologia proposta. Per verificare se l'algoritmo ha delle buone prestazioni è necessario confrontare i risultati ottenuti con i risultati forniti da altre metodologie presenti in letteratura. In particolare, tra tutte le metodologie esistenti, è stata presa in considerazione quella più performante, ovvero quella utilizzata dal tool FormalM [6].

5.3.1 FormalM

Questa sezione contiene un breve excursusu sulla metodologia utilizzata dal FormalM.

L'algoritmo di inferenza considerato prende in ingresso un grafo diretto completamente connesso, ogni coppia di vertici è collegata da due archi univoci (uno per ogni direzione), contenente n vertici, rappresentativi dei geni, ed

n^2 archi, rappresentativi delle connessioni esistenti tra i geni. Analogamente alla nostra metodologia l'approccio permette la distizione tra archi attivatori ed archi inibitori. il tool prende in ingresso le timeseries continue, le quali vengono discretizzate e successivamente utilizzate per produrre un modello CCS, Calculus of Communicating Systems [16], che descrive l'evoluzione dell'espressione genica nel tempo. Per ogni arco del grafo crea un 'processo arco' del tipo $P_e = p_1 + \cdots + p_k$, dove ogni P_i è un 'processo perturbazione' con i che indica l'i-esima perturbazione. L'operatore + evidenzia l'indipendenza tra le k perturbazioni prese in considerazione. Ogni singolo processo p_i è modellato come una sequenza di stati in cui la rete genica può trovarsi; i possibili stati discretizzati sono: UP, BASAL, DOWN.

Il modello CCS è utilizzato per eliminare dal grafo completo tutti gli archi che non soddisfano determinate proprietà, espresse mediante logica temporale. La verifica formale viene effettuata utilizzando la tecnica di verifica attraverso model checking basato sul mu-calculus. L'approccio considera due proprietà, che rispecchiano le funzioni biologiche fondamentali che possono verificarsi tra una coppia di geni: *attivazione* e *inibizione*. Le due proprietà verificano quanto segue:

- **arco attivatore**: ogni qual volta X è up, il gene Y deve diventare up e non deve ritornare più down;
- **arco inibitore**: ogni qual volta X è up, il gene Y deve diventare down e non deve ritornare più up.

Le principali differenze tra le 2 metodologie risiedono principalmente nella verifica formale tramite model checking. Infatti il nostro algoritmo prende utilizza un model cheker probabilistico.

Confronto di FormalM con altri tool

In questa sezione sono riportate le prestazioni del tool FormalM confrontate con quelle di altri tool presenti in letteratura, quali: ARACNE [15], CRL [9], TD-Aracne [24], TSNIF [19] e BANJO [22]. Questi tool, come FormalM, prendono in ingresso i dati relativi all'espressione dei geni nel tempo (le timeseries) e li elaborano per ricostruire la rete di regolazione genica.

ARACNE e TD-Aracne (Algorithm for the Reconstruction of Accurate Cellular Networks) utilizzano un approccio basato sulla teoria dell'informazione pereliminare, la stragrande maggioranza di interazioni indirette tipicamente inferite da un'analisi a coppie.

CLR (Context Likelihood of Relatedness) impiega uno stimatore sensibile al modo in cui i geni sono relazionati ed in modo particolare va a trasformare la stima dell'informazione mutua tra coppie di geni in un valore numerico (z-score), ottenendo dati standardizzati (media nulla e varianza unitaria).

BANJO (Bayesian Network Inference with Java Objects) si basa sulle reti Bayesiane con una struttura di inferenza basata sul punteggio. Per costruire la rete genica utilizza algoritmi di ricerca basati su euristiche (greedy hill-climbing).

TSNIF (Time Series Network Identification) descrive la rete di regolazione come un sistema di equazioni differenziali lineari che rappresentano la velocità di sintesi di una trascrizione in funzione delle concentrazioni di ogni altra trascrizione in una cellula.

Gli sviluppatori di FormalM hanno effettuato un confronto con tutti questi altri tool elencati e i risultati che hanno ottenuto sono riportati in figura5.8. Il grafico relaziona il valore dell'F-Measure all'aumentare del numero dei geni presenti nelle reti considerate.

Come si può notare le prestazioni del FormalM sono risultate essere le

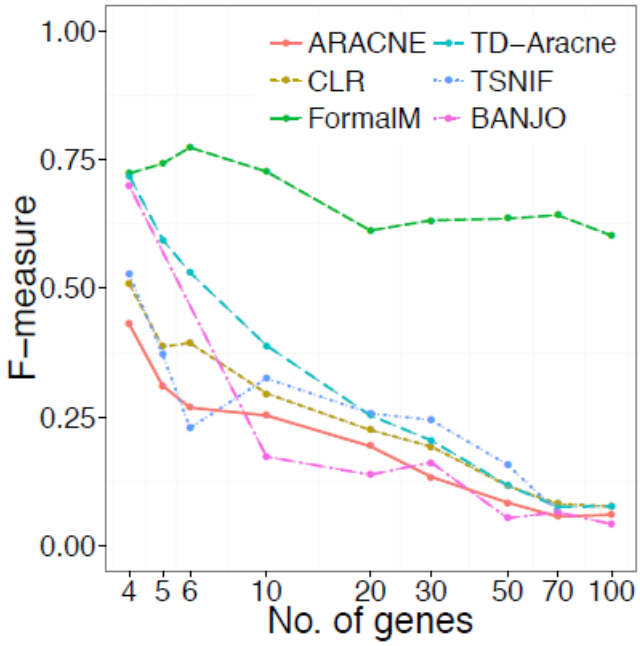

Figura 5.8: Performance di FormalM rispetto ad altri tool.

migliori. Infatti all'aumentare del numero di geni i valori dell'F-Measure per gli altri algoritmi calano drasticamente, mentre quelli ottenuti dal FormalM, seppur calino, restano ugualmente alti.

Data l'elevata accuratezza e precisione di questo tool si è deciso di utilizzarlo come termine di paragone per meglio valutare le prestazioni del nostro algoritmo.

5.3.2 Confronto dei Risultati

Al fine di ottenere un confronto omogeneo, sono state prese in considerazione le stesse reti utilizzate nelle sezioni precedenti per validare la metodologia proposta. Per una maggiore leggibilità dei dati sono stati considerati solo i valori medi di: Precizion, Recall ed F-Measure. Nella figura 5.9 è riportato in confronto tra le due metodologie in termi di precision. Il grafico riporta sulle

ascisse la dimensione della rete, ovvero il numero di geni presenti in esse, e sulle ordinate il rispettivo valre della Precision.

Osservando il grafico si evince che la precision è pressoché identica nel caso di reti aventi 4 e 6 geni. La metogologia proposta ha una precision maggiore in reti che hanno 5 geni. Per gli ultimi due casi invece, la precision del nostro algoritmo è inferiore a quella ottenuta dal FormalM. Nel caso di reti aventi 15 geni, però, la differenza tra i valori delle precision è minima.

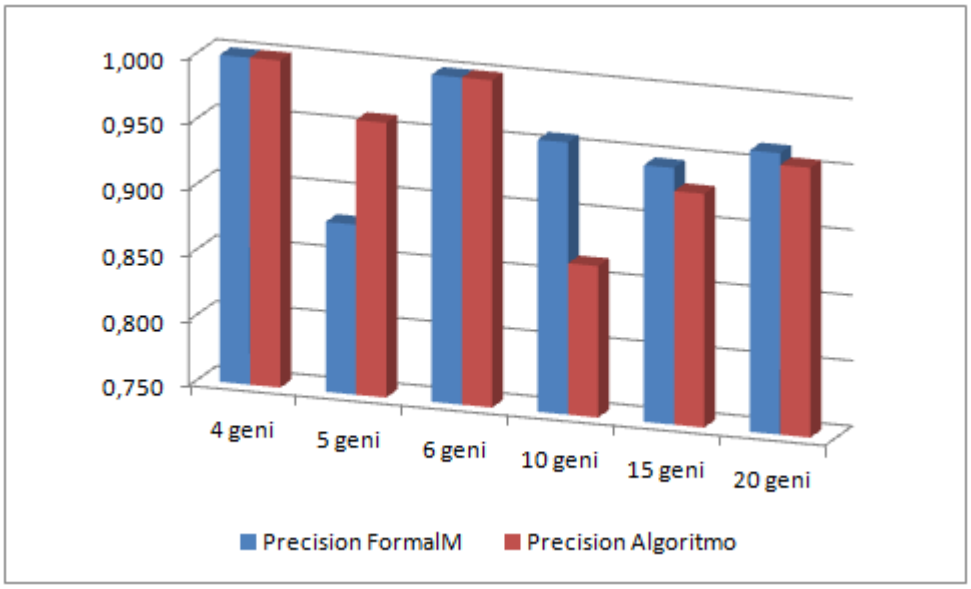

Figura 5.9: Confronto risultati ottenuti in termini di Precision.

Passiamo adesso a confrontare i risultati ottenuti in termini di recall, focalizzando l'attenzione sulla completezza delle due metodologie.

Dal grafico in figura 5.10 possiamo subito notare che la differenza tra i valori della recall è minima. Nella maggior parte dei casi, seppur minimamente, il valore della recall risultante dal nostro algoritmo è maggiore rispetto a quella ottenuta con il tool FormalM.

Per meglio valutare l'indice di bontà dei due algoritmi è stato confrontato il valore dell'F-Measure. Da questo confronto di questo indice abbiamo potuto

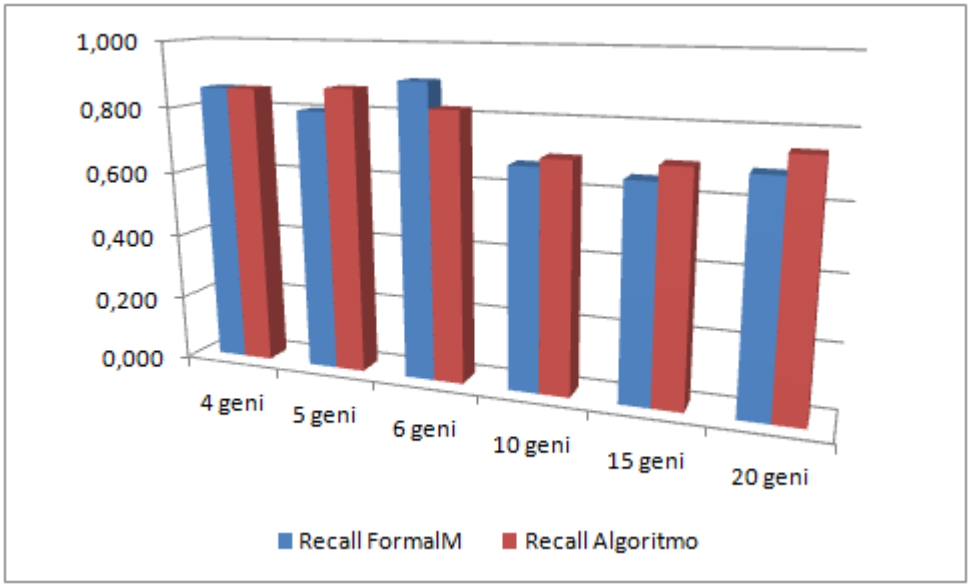

Figura 5.10: Confronto risultati ottenuti in termini di Recall.

individuare lalgoritmo migliore.

Osservando i risultati del confronto sono riportati in figura 5.11 è possibile affermare che la metodologia proposta mostra un F-Measure maggiore in 3 casi su 5. In un caso i valori sono identici; l'unico caso in cui FormalM risulta essere migliore è quello di reti aventi sei geni.

In definitiva è possibile affermare che l'algoritmo proposto presenta delle migliorie. Per meglio riassumere le differenze tra i due algoritmi è stato prodotto il grafico 5.12, che visualizza l'andamento del valore dell'f-measure all'aumentare del numero di geni.

Osservando il grafico si può subito notare che le prestazioni tendono a calare all'aumentare del numero di geni. Il valore dell'f-measure nel caso del nostro algoritmo non scende mai al di sotto di 0.75, e questo vuol dire che le prestazioni, in valore percentuale, sono sempre maggiori del 75%.

Purtroppo per quanto riguarda il tempo di esecuzione il nostro algoritmo è molto più lento rispetto al FormalM. Quindi in definitiva possiamo dire che

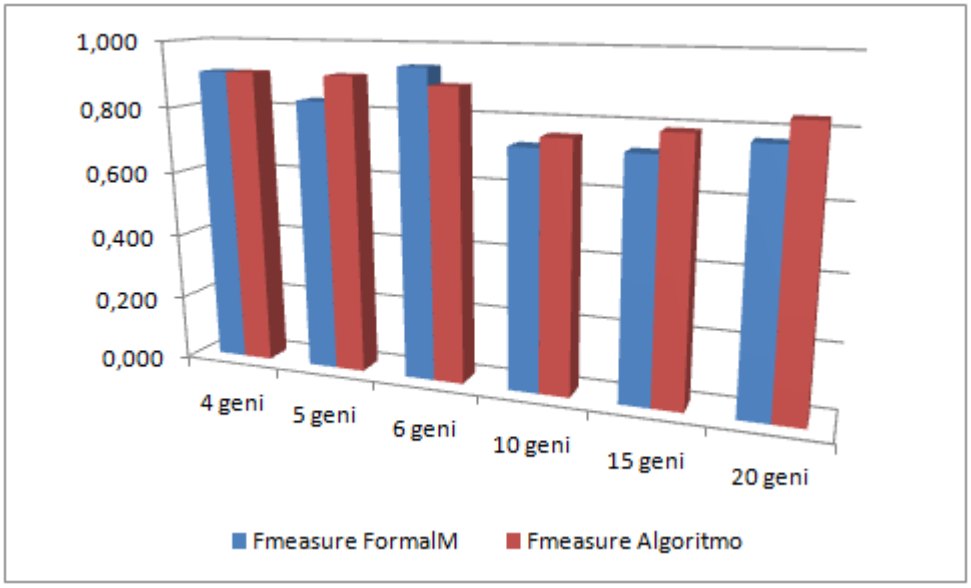

Figura 5.11: Confronto risultati ottenuti in termini di F-Measure.

in quanto a prestazioni il nostro algoritmo riporta i risultati migliori però li ottiene in un tempo maggiore. La scelta dell'algoritmo ottimale non può, quindi essere fatta poiché se si volessero ottenere dei risultati migliori comunque si dovrebbe aspettare un tempo più lungo. Il miglioramento in termini di performance si paga in maggiore tempo di esecuzione.

La scelta del tool migliore viene lasciata all'utilizzatore: all'utente finale viene lasciato l'onere della scelta, ovvero sta a lui decidere se vuole ottenere i risultati migliori oppure vuole ottenere dei risultati in breve tempo. In definitiva le prestazioni del tool FormalM sono comunque ugualmente buone, infatti le sue performance non scendono mai al di sotto del 70%.

Un'altra miglioria proposta dal nostro algoritmo è la possibilità di dare in pasto al tool più di una rete alla volta, sarà lui successivamente a gestire l'input andando a trovare i file appartenenti alla stessa rete. Il formalM, invece, permette solo l'esecuzione di una rete per volta: per passare alla rete successiva occorre nuovamente dargli i file in input. Questo risultato è stato ottenuto

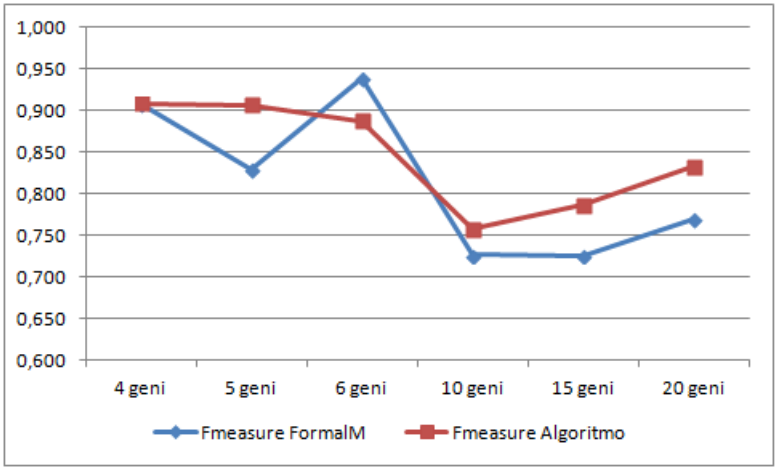

Figura 5.12: F-Measure media a confronto.

semplicemente dando in ingresso al tool non i singoli file ma una cartella che li contiene tutti e, successivamente, il tool seleziona i file appartenenti alla stessa rete e li elabora.

Un'ultima differenza sta nel tipo di file che i due tool prendono in ingresso, infatti FormalM oltre al file delle timeseries discretizzate prende in ingresso anche il file delle timeseries contenente i valori a tempo continuo. Nel nostro algoritmo, invece, sono sufficienti solo i dati discretizzati. Questa miglioria è stata ottenuta andando ad infoltire il numero di controlli fatti sui dati; infatti come precedentemente accennato nel capitolo della stesura del modello, oltre a verificare le proprietà che individuano la presenza dell'arco, vengono introdotti altri controlli che danno ulteriori informazioni sulla rete in esame.

Capitolo 6

Conclusioni e Sviluppi Futuri

La modellizzazione di sistemi biologici é un passo cruciale nella comprensione del loro funzionamento e nelle applicazioni quali il controllo di tali sistemi e l'ingegnerizzazione di nuove funzioni biologiche.

In questo lavoro di tesi, dopo aver inquadrato il problema dal punto di vista biologico, ci siamo occupati della definizione di una metodologia per l'identificazione di reti genetiche.

Inizialmente é stato definito un modello che simula il comportamento della rete genica. Questo modello é stato creato mediante l'ausilio di metodi formali.

A valle della creazione del modello sono state definite delle proprietá al fine di effettuare una verifica formale dello stesso; questa verifica é basata sul model checking. Le proprietá identificate sono state ricavate a partire da un'analisi preliminare dei dati a disposizione e ispirandosi a quanto giá presente in letteratura.

I risultati ottenuti dalla verifica formale sono stati elaborati mediante un algoritmo creato ad hoc, il quale restituisce, sotto forma di 'archi', le relazioni che intercorrono tra i geni della rete, ovvero il modo in cui questi ultimi interagiscono tra di loro.

Infine sono state valutate le prestazioni dell'algoritmo andando ad analizzare i valori di Precision, Recall ed F-Measure. Per poter calcolare suddette metriche sono stati eseguiti dei test, in particolare sono state mandate in ingresso al tool reti contenenti un diverso numero di geni; siamo partiti da reti piccole contenenti 4 geni fino ad arrivare a considerare reti con un massimo di geni pari a 20. Per ogni tipologia di rete individiduata sono state considerate 10 reti differenti. I risultati di ciascuna rete, in termini di archi individuati, sono stati utilizzati per individuare i veri positivi, i falsi positivi ed i falsi negativi. Una combinazione di suddetti parametri ci ha permesso di determinare, per ciascuna rete analizzata, Precision, Recall ed F-measure. In una prima fase abbiamo considerato i risultati nel complesso per poter valutare la bontá dell'algoritmo al variare del numero di geni appartenenti alle reti prese in considerazione.

In una seconda fase si é pensato di valutare le prestazioni della metodologia proposta con quelle fornite da uno dei tools presenti in letteratura; in particolare abbiamo scelto di utilizzare il tool che presenta le caratteristiche piú performanti (FormalM). Al fine di garantire un confronto omogeneo le reti considerate in questo seconda fase sono perfettamente identiche a quelle utilizzate per la valutazione della fase precedente. I risultati ottenuti in entrambi i casi sono stati valutati in termini di valori medi, ovvero per ciascuna tipologia di rete abbiamo mediato i valori di Precision, Recall ed F-measure rispettivamente sui risultati ottenuti dall'analisi delle 10 reti. Confrontando i risultati cosí ottenuti é emerso che l'algoritmo prodotto ha delle buone prestazioni. Infatti i risultati sono del tutto paragonabili a quelli dei tool giá presenti in letteratura e nella maggior parte dei casi si evince la presenza di miglioramenti, seppur minimi.

Va sottolineato che l'algoritmo prodotto ha dei tempi di esecuzione, che

crescono all'aumentare del numero di geni appartenenti alle reti analizzate, maggiori di quelli mostrati dal tool FormalM. Quanto detto non ci permette di stabilire in assoluto l'algoritmo che presenta le migliori prestazioni. In modo particolare, se l'utilizzatore é interessato a parametri quali accuratezza e completezza il nostro algoritmo puó essere ritenuto il migliore, dal momento che la miglioria, seppur minima é presente. D'altronde, se l'utilizzatore volesse utilizzare un tool in grado di fornire dei risultati in tempi relativamente brevi potrebbe pensare di utilizzare l'algoritmo FormalM, pagando un pó in termini di accuratezza e completezza. Pertanto, in conclusione possiamo affermare che la sceltadella metodologia da utilizzare é prettamente a discrezione dell'utilizzatore.

Concludendo possiamo dire che siamo soddisfatti della metodoligia prodotta e siamo convinti che possa essere migliorata ulteriormente.

Nella prossima sezione verranno dettagliate alcuni possibili sviluppi futuri.

6.1 Sviluppi Futuri

In base a quanto affermato nella sezione inerente le conclusioni sul lavoro svolto é possibile affermare che, affinché la metodologia individuata sia effettivamente migliore in termini di prestazioni (tempi di esecuzione, accuratezza e completezza) é necessario considerare l'introduzione di nuove 'feature'.

Dal momento che l'aspetto maggiormente critico risulta essere quello associato ai tempi di esecuzione alquanto elevati si potrebbe pensare, come primo sviluppo futuro, ad un miglioramento di questi ultimi. In particolare si potrebbe cercare di ridurre il numero di controlli effettuati dal model checker (PRISM). Questa considerazione scaturisce dal fatto che é stato notato come il degrado dei tempi di esecuzione sia legato principalmente al tool PRISM.

Per diminuire il numero delle proprietá da verificare per ogni rete si potrebbe pensare di effettuare un'ulteriore analisi preliminare dei dati.

Raggiunto suddetto obiettivo si potrebbero andare a valutare le prestazioni della metodologia sviluppata su reti con un numero di geni superiore a 20. In particolare si potrebbe pensare di eseguire l'algoritmo ponendo in input reti con rispettivamente 50 e 100 geni. Questo aspetto ci permetterÃ di confrontare in modo esaustivo la metodologia con il tool FormalM [1].

Affinché sia possibile migliorare la veridicitá dei risultati ottenuti si potrebbe pensare di mdificare la metodologia andando a ridurre il numero dei falsi positivi e dei falsi negativi. In questo modo si otterrebbe un miglioramento sia della Precision che della Recall e, conseguentemente, dell'F-measure.

Una possibile miglioria si potrebbe ottenere andando ad individuare ulteriori proprietá e, successivamente, ad introdurle al fine di individuare meglio alcuni particolari archi della rete (ad esempio gli archi che hanno a destra del segno di relazione un gene (x) che inizialmente viene attivato da un gene y e successivamente viene inibito da un altro gene (z)). Cosí facendo si potrebbe ridurre il numero di falsi negativi andando a migliorare la Recall.

Per ridurre il numero di falsi positivi, invece, e quindi migliorare la Precision, si potrebbe introdurre un ulteriore controllo sui risultati che la metodologia ci fornisce in output. In particolare, come nel caso precedente, si potrebbero introdurre ulteriori proprietá che verificano la veridicitá dei risultati ottenuti. In altre parole, bisognerebbe validare ogni arco presente nel file di output in termini di appartenenza effettiva o meno alla rete genica reale.

[1] Da quanto emerso in letteratura il tool FormalM ha delle buone prestazioni fintanto che si considerano reti con un numero massimo di geni pari a 100.

Appendice A

Risultati FormalM

Di seguito sono riportati le tabelle riassuntive contenenti i risultati di Precision, Recall ed F-Measure calcolati per il FormalM. Anche in questo caso sono state utilizzate reti che avevano un numero crescente di geni e per ogni tipologia sono state scelte dieci reti differenti. Le reti utilizzate sono le stesse date in ingresso al nostro algoritmo.

RETI 4 GENI FormalM						
Nome Rete	**VP**	**FP**	**FN**	**Precision**	**Recall**	**Fmeasure**
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	2	0	2	1,000	0,500	0,667
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	2	0	2	1,000	0,500	0,667
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	3	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	3	0	2	1,000	0,600	0,750

Figura A.1: Risultati ottenuti su una rete di 4 geni.

RETI 5 GENI FormalM						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	3	0	2	1,000	0,600	0,750
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	0	4	4	0,000	0,000	0,000
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	4	1	1	0,800	0,800	0,800
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	3	0	2	1,000	0,600	0,750
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	4	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	4	0	0	1,000	1,000	1,000

Figura A.2: Risultati ottenuti su una rete di 5 geni.

RETI 6 GENI FormalM						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	3	0	2	1,000	0,600	0,750
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	4	0	2	1,000	0,667	0,800
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	5	0	2	1,000	0,714	0,833
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	5	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	7	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	5	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	5	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	7	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	5	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	5	0	0	1,000	1,000	1,000

Figura A.3: Risultati ottenuti su una rete di 6 geni.

RETI 10 GENI FormalM						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	8	0	9	1,000	0,471	0,640
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	1	0	8	1,000	0,111	0,200
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	11	4	0	0,733	1,000	0,846
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	8	0	1	1,000	0,889	0,941
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	11	2	8	0,846	0,579	0,688
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	6	0	3	1,000	0,667	0,800
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	9	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	9	0	1	1,000	0,900	0,947
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	1	0	8	1,000	0,111	0,200
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	16	0	0	1,000	1,000	1,000

Figura A.4: Risultati ottenuti su una rete di 10 geni.

RETI 15 GENI FormalM						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	14	0	0	1,000	1,000	1,000
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	1	0	13	1,000	0,071	0,133
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	13	0	1	1,000	0,929	0,963
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	24	0	2	1,000	0,923	0,960
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	11	7	7	0,611	0,611	0,611
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	5	0	11	1,000	0,313	0,476
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	14	1	14	0,933	0,500	0,651
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	11	1	6	0,917	0,647	0,759
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	19	0	14	1,000	0,576	0,731
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	20	0	1	1,000	0,952	0,976

Figura A.5: Risultati ottenuti su una rete di 15 geni.

RETI 20 GENI FormalM						
Nome Rete	VP	FP	FN	Precision	Recall	Fmeasure
ecoli_transcriptional_network_regulonDB_6_7-1_dream4	20	0	1	1,000	0,952	0,976
ecoli_transcriptional_network_regulonDB_6_7-2_dream4	18	1	9	0,947	0,667	0,783
ecoli_transcriptional_network_regulonDB_6_7-3_dream4	1	0	18	1,000	0,053	0,100
ecoli_transcriptional_network_regulonDB_6_7-4_dream4	19	0	9	1,000	0,679	0,809
ecoli_transcriptional_network_regulonDB_6_7-5_dream4	19	0	7	1,000	0,731	0,844
ecoli_transcriptional_network_regulonDB_6_7-6_dream4	27	8	30	0,771	0,474	0,587
ecoli_transcriptional_network_regulonDB_6_7-7_dream4	19	0	7	1,000	0,731	0,844
ecoli_transcriptional_network_regulonDB_6_7-8_dream4	18	0	1	1,000	0,947	0,973
ecoli_transcriptional_network_regulonDB_6_7-9_dream4	29	2	12	0,935	0,707	0,806
ecoli_transcriptional_network_regulonDB_6_7-10_dream4	29	0	2	1,000	0,935	0,967

Figura A.6: Risultati ottenuti su una rete di 20 geni.

Bibliografia

[1] Rajeev Alur and ThomasA. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[2] Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego Di Bernardo. How to infer gene networks from expression profiles. *Molecular systems biology*, 3(1), 2007.

[3] Mukesh Bansal and Andrea Califano. Genome-wide dissection of post-transcriptional and posttranslational interactions. In *Gene Regulatory Networks*, pages 131–149. Springer, 2012.

[4] Joel R Bock and David A Gough. Predicting protein–protein interactions from primary structure. *Bioinformatics*, 17(5):455–460, 2001.

[5] Michele Ceccarelli, Luigi Cerulo, and Antonella Santone. De novo reconstruction of gene regulatory networks from time series data, an approach based on formal methods. *Methods*, 69(3):298–305, 2014.

[6] Michele Ceccarelli, Luigi Cerulo, and Antonella Santone. De novo reconstruction of gene regulatory networks from time series data, an approach based on formal methods. *Methods*, 69(3):298–305, 2014.

[7] Luigi Cerulo, Charles Elkan, and Michele Ceccarelli. Learning gene regulatory networks from only positive and unlabeled data. *Bmc Bioinformatics*, 11(1):228, 2010.

[8] Luigi Cerulo, Vincenzo Paduano, Pietro Zoppoli, and Michele Ceccarelli. A negative selection heuristic to predict new transcriptional targets. *BMC bioinformatics*, 14(Suppl 1):S3, 2013.

[9] Jeremiah J Faith, Boris Hayete, Joshua T Thaden, Ilaria Mogno, Jamey Wierzbowski, Guillaume Cottarel, Simon Kasif, James J Collins, and Timothy S Gardner. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS biology*, 5(1):e8, 2007.

[10] Timothy S. Gardner and Jeremiah J. Faith. Reverse-engineering transcription control networks. *Physics of Life Reviews*, 2(1):65 – 88, 2005.

[11] Hendrik Hache, Hans Lehrach, and Ralf Herwig. Reverse engineering of gene regulatory networks: a comparative study. *EURASIP Journal on Bioinformatics and Systems Biology*, 2009:8, 2009.

[12] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[13] Yong Li, Lili Liu, Xi Bai, Hua Cai, Wei Ji, Dianjing Guo, and Yanming Zhu. Comparative study of discretization methods of microarray data for inferring transcriptional regulatory networks. *BMC bioinformatics*, 11(1):520, 2010.

[14] Shoudan Liang, Stefanie Fuhrman, Roland Somogyi, et al. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific symposium on biocomputing*, volume 3, pages 18–29, 1998.

[15] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo D Favera, and Andrea Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC bioinformatics*, 7(Suppl 1):S7, 2006.

[16] Robin Milner. *Communication and concurrency*. Prentice-Hall, Inc., 1989.

[17] Fantine Mordelet and Jean-Philippe Vert. Sirene: supervised inference of regulatory networks. *Bioinformatics*, 24(16):i76–i82, 2008.

[18] Sandro Morganella, Pietro Zoppoli, and Michele Ceccarelli. Iris: a method for reverse engineering of regulatory relations in gene networks. *BMC bioinformatics*, 10(1):444, 2009.

[19] Athanasios Polynikis, SJ Hogan, and Mario di Bernardo. Comparing different ode modelling approaches for gene regulatory networks. *Journal of theoretical biology*, 261(4):511–530, 2009.

[20] Thomas Schaffter, Daniel Marbach, and Dario Floreano. GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, 2011. wingx.

[21] Jean-Philippe Vert. Reconstruction of biological networks by supervised machine learning approaches. *Elements of Computational Systems Biology*, pages 165–188, 2010.

[22] Jing Yu, V Anne Smith, Paul P Wang, Alexander J Hartemink, and Erich D Jarvis. Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, 2004.

[23] Xiujun Zhang, Xing-Ming Zhao, Kun He, Le Lu, Yongwei Cao, Jingdong Liu, Jin-Kao Hao, Zhi-Ping Liu, and Luonan Chen. Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information. *Bioinformatics*, 28(1):98–104, 2012.

[24] Pietro Zoppoli, Sandro Morganella, and Michele Ceccarelli. Timedelay-aracne: Reverse engineering of gene networks from time-course data by an information theoretic approach. *Bmc Bioinformatics*, 11(1):154, 2010.