Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Specialistica in Ingegneria Informatica

Ragionamento Qualitativo in Genomica

Laureando:
Daniele Vaccari

Relatore:
Ch.ma Prof. Silvana Badaloni
Correlatore:
Dott. Francesco Sambo

Anno accademico 2009/2010

Introduzione

L'uomo si è sempre interrogato sulla natura della vita e sulle sue particolarità. Lo spettacolo dell'enorme varietà degli organismi viventi sulla Terra non ha mancato di stupirlo e incuriosirlo, come pure la miracolosa capacità della vita di ripararsi e di rigenerare se stessa. In ogni cellula del nostro corpo, come di quello di ogni altro essere vivente, sono racchiuse le istruzioni per crescere, per mantenersi in vita, giorno dopo giorno, e per generare a suo tempo una prole.
La struttura e le attività di una cellula o di un organismo pluricellulare dipendono, a livello molecolare, dalla presenza di determinate proteine e dal numero di molecole di ciascuna proteina. Variazioni in queste due caratteristiche consentono di adeguare le attività metaboliche alle necessità del momento in ogni fase della vita, rendendo perciò gli organismi adatti all'ambiente che li circonda.
Come afferma il *dogma centrale della biologia molecolare*[18], l'informazione genetica è conservata nel DNA, che viene trascritto sotto forma di RNA, il quale viene successivamente tradotto in proteine, la forma operativa dell'informazione contenuta nel genoma. Questo costituisce il meccanismo di base dell'espressione dei geni e la direzione fondamentale del flusso di informazione genetica. La regolazione dell'espressione genica è fondamentale per la cellula, perchè le permette di controllare le proprie funzioni interne ed esterne, come la differenziazione cellulare, la morfogenesi o i vari processi di adattamento alle necessità dell'organismo. Alcune proteine, chiamate *fattori di trascrizione*, hanno il compito, in combinazione con altre, di attivare o inibire la trascrizione dei geni e di controllare la trasformazione dell'RNA in nuove proteine. Questo processo è chiamato *regolazione genica*.
Il termine *reverse engineering* indica l'insieme dei metodi, utili a ricostruire la complessa rete di regolazione e controllo dell'output dinamico del sistema osservato. Data la complessità del sistema analizzato, gli approcci di reverse engineering per lo studio di interazione tra geni, proteine ed altri metaboliti è in genere limitato a piccole parti della rete di regolazione. Spesso in questi casi la rete di regolazione che si vuole studiare è già parzialmente nota, in alcuni casi a livello di dettaglio biochimico, e lo studio, così come la scelta dei geni e delle proteine da monitorare è "hypotesis driven", si basa cioè sulla conoscenza a priori e sulle ipotesi scentifiche che si voglio verificare.

L'avvento delle tecnologie high-throughput per lo studio di sistemi biologici, come ad esempio i DNA-microarray[20] o la spettrometria di massa[17], ha dato la possibilità di passare da un approccio "hypothesis driven" ad un approccio "question driven", in cui viene analizzato il sistema nel suo complesso invece di scegliere a priori le molecole da monitorare.
In genere è possibile monitorare l'intero genoma (migliaia o decine di migliaia di trascritti) a fronte di qualche decina o centinaia di array (sia per motivi tecnici ed etici legati alla raccolta dei campioni, che per ragioni di costo). Pertanto i modelli di regolazione applicati a questo tipo di dati non possono essere molto complessi e includere un elevato numero di parametri[7]. Inoltre, attraverso le tecnologie high-throughtput non è possibile monitorare tutti i diversi tipi di molecole che partecipano al processo di regolazione e controllo, pertanto i modelli sono di fatto semplificati identificando il livello di espressione dell'RNA come un'approssimazione del livello di espressione proteica delle proteine codificate dall'RNA stesso. Una rete di regolazione basata su quest'ultimo modello è detta rete di regolazione genica e può essere rappresentata come un grafo diretto in cui i nodi rappresentano i geni o le proteine da essi codificate, mentre gli archi orientati rappresentano l'azione regolatoria tra nodi diversi.
Obbiettivo principale del presente lavoro di tesi è l'introduzione della gestione del segno delle regolazioni nell'algoritmo di Ragionamento Qualitativo sviluppato da F. Sambo [30] che, partendo da un confronto qualitativo tra i profili d'espressione dei geni, identifica possibili relazioni causa-effetto tra coppie di geni.
Il Ragionamento Qualitativo è un approccio tipico dell'Intelligenza Artificiale, nel quale alle variabili numeriche quantitative vengono preferiti dei valori simbolici qualitativi, che, ad esempio identificano l'assenza o la presenza di un gene in un determinato esperimento piuttosto del valore numerico del livello d'espressione del gene.
Si è visto che approcci di questo tipo forniscono risultati migliori per studi di DNA-microarray, rispetto agli studi classici basati su elaborazioni statistiche e matematiche.
Come si vedrà in seguito, le estensioni apportate al framework di partenza sono state numerose e hanno permesso di ottenere notevoli miglioramenti, sia per quel che riguarda l'usabilità dello strumento, ma soprattutto per quanto riguarda le regolazioni inferite: in numero maggiore e con alti livelli di precisione.
Nel capitolo 1 verrà innanzitutto introdotto il problema biologico che sta alla base della tesi. Nel capitolo 2 si discuterà dello stato dell'arte e in particolare dell'algoritmo usato come framework di partenza. Verranno, inoltre, mostrati i dataset utilizzati per testare e validare i risultati ottenuti dall'algortimo e gli indici utilizzati per misurare le prestazioni.
Nel capitolo 3 verranno discusse le estensioni e le nuove funzionalità introdotte nel framework di partenza. Infine, nel capitolo 4 verranno mostrati i

risultati ottenuti dall'algoritmo e confrontati con quelli ottenuti dall'algoritmo di partenza. Infine nel capitolo 5 verranno tratte alcune considerazioni sullo studio condotto ed indicati possibili sviluppi futuri.

Indice

Capitolo 1

Il Problema Biologico

Il genoma può essere considerato come l'insieme di tutte le istruzioni utili ad ogni cellula del nostro organismo per codificare le proteine necessarie alla sopravvivenza, all'interazione con l'ambiente e alla riproduzione, quindi in ultima analisi, alla vita stessa. Negli ultimi 10 anni sono stati sequenziati centinaia di genomi di organismi diversi. In particolare, il sequenziamento del genoma umano[25], cioè la determinazione dell'ordine dei diversi nucleotidi che costituiscono il DNA, è stato tra i traguardi più promettenti della biologia molecolare ed ha aperto la strada alla cosiddetta era post genomica. Ogni cellula del nostro organismo contiene l'intera sequenza del genoma che ci caratterizza come individui; ciò che le differenzia è una diversa espressione proteica, vale a dire il bagaglio delle proteine espresse o esprimibili in risposta ad uno stimolo, che permette alle diverse cellule di assumere caratteristiche e compiti specifici e differenziati. In effetti, 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.
In questo capitolo verranno trattati gli aspetti biologici legati ai meccanismi di regolazione e controllo di quest'informazione, i cui due passi principali sono la trascrizione, durante la quale il DNA è trascritto in RNA, e la traduzione, durante la quale l'RNA è tradotto in proteina[18].

1.1 Regolazione Genica

Il processo di trasferimento dell'informazione genetica, a partire dal passaggio della trascrizione del DNA ad RNA, fino alle modificazioni post-trascrizionali della proteina prodotta è fondamentale per la cellula, perchè le permette di controllare le proprie funzioni interne ed esterne, come la differenziazione cellulare, la morfogenesi o i vari processi di adattamento alle necessità dell'organismo.
Si presenteranno ora le componenti biologiche che intervengono in questo

processo, descrivendo sinteticamente anche le fasi della regolazione genica citate pocanzi.

1.1.1 DNA, RNA e Proteine

Nel 1931 P.A. Levene[2] ha dimostrato che gli acidi nucleici sono costituiti da grosse molecole che per idrolisi totale producono unità più semplici dette **nucleotidi**. Ogni nucleotide è formato da tre componenti: un gruppo fosfato, uno zucchero pentoso e una base azotata. Le basi azotate possibili sono cinque, suddivisibili in due categorie: purine e pirimidine. Le basi puriniche sono l'adenina (A) e la guanina (G), mentre le basi pirimidiniche sono la citosina (C), la timina (T) e l'uracile (U). Esistono due tipi di acidi nucleici: il DNA (*DeoxyriboNucleic Acid*, acido deossiribonucleico) e l'RNA (*RiboNucleic Acid*, acido ribonucleico). Un'altra differenza fondamentale tra i due tipi di acidi nucleici riguarda le basi azotate contenute: nel DNA possono trovarsi quattro diversi nucleotidi che differiscono soltanto per la base azotata, così come nell'RNA. Tuttavia, la timina è presente soltanto nel DNA e al suo posto nell'RNA è presente l'uracile.
Il DNA è localizzato prevalentemente nel nucleo e rappresenta il costituente dei geni e quindi dei cromosomi, mentre l'RNA è concentrato nel citoplasma soprattutto in corrispondenza dei ribosomi.
Watson e Crick[3] proposero che il DNA avesse una struttura a forma di doppia elica destrorsa, con le due catene polinucleotidiche antiparallele avvolte l'una sull'altra in una spirale.
Ogni filamento di DNA è un polinucleotide, contiene cioè tanti nucleotidi uniti tra loro da legami covalenti secondo una disposizione lineare. L'appaiammento tra le basi è molto specifico poichè l'adenina (A) è sempre unita alla timina (T) da due legami idrogeno e la guanina (G) è sempre unita alla citosina (C) da tre legami idrogeno. Quindi, in un qualsiasi punto della molecola si può avere tra coppie di basi complementari un legame del tipo: A=T o T=A e G≡C o C≡G. Conseguentemente, la successione delle basi di un filamento condiziona la successione delle basi dell'altro filamento. A differenza del DNA, l'RNA è costituito da un singolo filamento composto da una catena polinucleotidica che può avere una lunghezza di parecchie cenetinaia o migliaia di unità elementari. La struttura primaria di un filamento di RNA è molto simile a quella di un filamento di DNA.
Oltre agli acidi nucleici, un altro tipo di macromolecole che merita di essere citato per il loro contenuto di informazione genetica sono le proteine o protidi. Esse rappresentano un ampio gruppo di composti organici formati da uno o più polipeptidi, cioè da catene di più amminoacidi legati tra loro attraverso legami peptidici, a volte accompagnati da altre molecole ausiliarie. Come si vedrà più avanti le proteine derivano dall'RNA, infatti un amminoacido non è altro che l'unione di tre o più nucleotidi. Generalmente si definisce *codone* una tripletta di nucleotidi e sono necessari uno o più codo-

ni per formare un'amminoacido. Ciò deriva dal fatto che i nucleotidi sono soltanto 4 mentre gli amminoacidi sono 20, quindi per codificare un amminoacido è necessaria l'informazione contenuta in almeno 3 nucleotidi. Tra i 20 amminoacidi esistenti, 8 sono classificati come essenziali in quanto l'organismo non è in grado di sintetizzare questi amminoacidi a partire da altri aminoacidi tramite trasformazioni biochimiche.
In base alla loro funzione le proteine possono essere distinte in: enzimi, di trasporto, contrattili, strutturali, di difesa e regolatrici.
Le proteine sono soggette ad un continuo processo di demolizione e sintesi, il turnover proteico, attraverso il quale l'organismo è in grado di rinnovare continuamente le proteine logorate sostituendole con nuovo materiale proteico.

1.1.2 I Geni

Si sa da tempo che il nostro patrimonio genetico può essere pensato come suddiviso in un certo numero di istruzioni di senso compiuto che sono state chiamate *geni*. Un gene è appunto un tratto di DNA di lunghezza molto variabile, da un migliaio a qualche centinaia di migliaia di nucleotidi, che porta scritta l'istruzione per compiere una determinata funzione biologica. A questa parola si è dato dapprima un significato molto ampio e anche piuttosto vago.
Un gene è necessario, per esempio, per avere una certa quantità di pigmento scuro, detto melanina, nella pelle, nei capelli e nella porzione pigmentata della retina. Se questo gene non funziona, infatti, si è albini. Un altro gene è necesario perchè possiamo distinguere bene i colori, per esempio il rosso e il verde. Ogni gene serve per compiere una determinata funzione: quando non presenta problemi non ci accorgiamo neppure della sua presenza. Soltanto quando non funziona a dovere siamo costretti a notare il suo operato perchè ci viene a mancare qualcosa. Storicamente, è questo il modo in cui ci siamo accorti dell'esistenza dei geni.

1.1.3 Dogma Centrale

Con la scoperta della struttura del DNA, l'osservazione che alcune proteine, enzimatiche e non, possano essere costituite da più polipeptidi ha portato ad affermare che un gene rappresenti una unità ereditaria funzionale costituita da una frazione della molecola del DNA che presiede alla sintesi di una particolare catena polipeptidica. Tale osservazione consentì, quindi, di passare alla nuova ipotesi **"un gene-una catena polipeptidica"**.
Le informazioni acquisite sul trasferimento dell'informazione ereditaria permisero di accertare che la relazione tra gene e carattere è mediata da una successione di eventi e di reazioni che hanno nel gene, costituito da una frazione di molecola di DNA, l'elemento primitivo dal quale prende avvio la

sintesi di una particolare catena polipetidica. Nel 1953, F. Crick[4] ipotizzò un processo a due fasi di espressione genica attraverso il quale l'informazione ereditaria contenuta in un dato gene poteva essere trasferita dalla sequenza di nucleotidi del DNA alla sequenza di nucleotidi dell'RNA e quindi alla sequenza di amminoacidi della proteina corrispondente.
I risultati di altri esperimenti confermarono la validità della sucessione ipotizzata: DNA → RNA → proteine. In seguito a queste acquisizioni risultò evidente che l'RNA è la molecola che funziona da mediatore mettendo in relazione l'informazione contenuta nella sequenza di basi del DNA con quella contenuta nella sequenza di amminoacidi delle proteine, rendendo così possibile il trasferimento dell'informazione genetica dai cromosomi ai ribosomi. La relazione tra DNA, RNA e proteine costituisce l'essenza di quello che viene definito *dogma centrale della biologia molecolare*[5]: **DNA → RNA → Proteina**, come esemplificato in Figura 1.1.

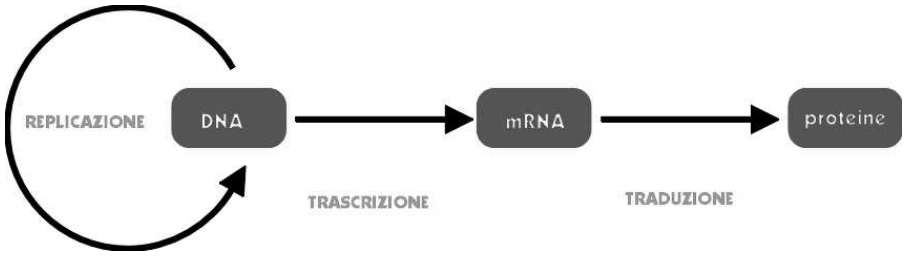


Fig. 1.1: Schema del dogma centrale della biologia molecolare.

Le molecole di RNA si formano sullo stampo del DNA - **trascrizione** - e l'RNA funge a sua volta da stampo per la sintesi delle proteine - **traduzione**. Ciò significa che l'informazione genetica presente nel DNA, sotto forma di una particolare sequenza di basi, viene prima copiata nell'RNA, impiegando come stampo uno dei due filamenti del DNA stesso, ed in seguito tradotta in catena polipeptidica, la cui composizione amminoacidica dipende dalla sequenza di basi trascritta nell'RNA. Tuttavia, questa successione di eventi non sempre è valida in quanto alcuni geni codificano unicamente per RNA che non sono traducibili in proteine.
Secondo questo modello, l'accoppiamento specifico delle basi azotate fornisce non solo un sistema per eseguire la copia esatta di ciascun gene durante la duplicazione del materiale ereditario (replicazione), ma costituisce anche il meccanismo alla base del trasferimento dell'informazione genetica da una molecola di DNA ad una di RNA (trascrizione). Successivamente l'RNA serve da stampo per disporre nella giusta sequenza gli amminoacidi nelle relative catene polipeptidiche (traduzione). Infine, esiste un sistema di controllo post-trascrizionale che regola le interazioni proteina-proteina e le modifiche post-trascrizionali, in altre parole tutta una serie di trasformazio-

ni e interazioni che possono modificare l'attività e la funzione della proteina e quindi, ad esempio, la sua capacità di regolare la trascrizione di altri geni. Queste particolari proteine che, in combinazione con altre, hanno il compito di attivare o inibire la trascrizione dei geni vengono chiamate *transcription factors* o fattori di trascrizione. Comunemente l'attivazione di un gene viene anche definita *positive regulation* (regolazione positiva), conseguentemente la *negative regulation* (regolazione negativa) identifica l'inibizione del gene.

1.2 DNA Microarray

Fra gli esperimenti di biologia molecolare che producono i datasets più voluminosi, i microarray sono tra i più importanti. Un insieme di esperimenti di microarray può descrivere contemporaneamente il livello di espressione di centinaia di migliaia di geni in centinaia di individui, fornendo la possibiltà di monitorare l'intero trascrittoma e traduttoma di un organismo/tessuto/cellula, in un dato istante e in una determinata condizione fisiologica. La tecnologia microarray per lo studio del profilo d'espressione genica è stata pubblicata per la prima volta nel 1995[8]. Il DNA-microarray è costituito da un supporto solido (vetro o plastica, materiale siliconico) dove sono immobilizzati delle molecole (*probes*) in modo ordinato mediante diversi metodi chimici e tecnologie di microdepositazione (*spotting*), supporto che verrà esposto ad una soluzione contenente la miscela di molecole da interrogare (*target*). La miscela di *targets* può essere costituita dall'insieme dei trascritti marcati con fluorocromi dopo trascrizione inversa da RNA totale per lo studio del *gene expression profiling* oppure dall'insieme di frammenti di PCR contenenti marcatori genici (SNP) per lo studio di SNP *mapping*.
Accennando brevemente alla tecnologia di fabbricazione dei microarray, essi possono essere fabbricati usando diverse tecnologie, ad esempio quelle che legano covalentemente la sonda (*probe*) usando maschere preformate da ditte specializzate e sintesi della sonda stessa sul supporto solido (*sintesi in situ*) oppure mediante microdeposizione di sonde gia presintetizzate (*sintesi off-line*).
Le misure ricavate da un esperimento con microarray possono essere rappresentate con una matrice di valori di espressione genica, le cui righe corrispondono ciascuna ad uno specifico trascritto monitorato e le colonne ai vari array. In genere è possibile monitorare l'intero genoma (migliaia o decine di migliaia di trascritti) a fronte di qualche decina o centinaia di array (sia per motivi tecnici ed etici legati alla raccolta dei campioni, che per ragioni di costo). Molto spesso gli esperimenti di DNA-microarray vengono ripetuti molteplici volte per ovviare al problema del rumore, molto presente in questa tipologia di analisi.
Uno studio condotto attraverso l'utilizzo di microarray prevede tipicamente l'esecuzione di molti esperimenti sullo stesso insieme di geni, atti a misurare

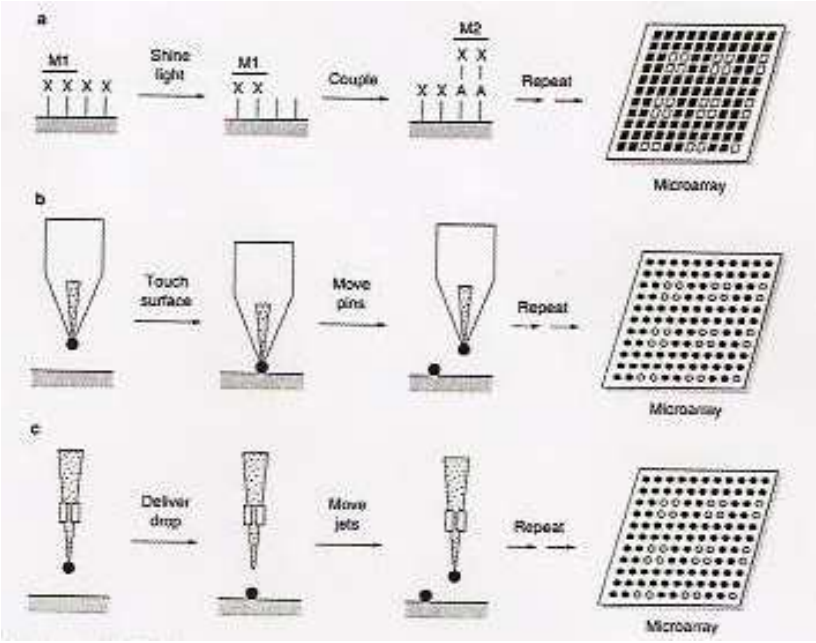


Fig. 1.2: Tecnologie di fabbricazione dei microarray:
a) fotolitografia; b) pin-spotting; c) sistema piezoelettrico inkject.

il profilo di espressione genica in vari istanti temporali o in condizioni di perturbazione differenti. Per quanto riguarda l'ambito temporale le misurazioni possibili sono essenzialmente di due tipi: *time courses*, o in evoluzione temporale, cioè per ogni istante che caratterizza l'evoluzione dell'esperimento e *steady state* ovvero al raggiungimento dello stato stazionario del sistema.
Come accennato precedentemente, gli esperimenti possono anche distinguersi in base alle presenza o meno di perturbazioni esterne applicate al sistema. Un'osservazione effettuata su un sistema privo di stimolazioni esterne e dunque in evoluzione libera viene definita *wild-type* e rappresenta generalmente il caso di riferimento per la rete formata dai geni in esame. Un esperimento con una singola regolazione esterna atta a inibire la trascrizione di un gene (down-regulation) viene detto esperimento di *gene knock-out*, inversamente un esperimento in cui un gene viene attivato mediante una perturbazione esterna (up-regulation), viene chiamato *over-expression*.
Le osservazioni ottenute mediante queste due tipologie di esperimento vengono in seguito confrontate con il wild-type, ricavando informazioni sugli effetti della perturbazione applicata.

1.3 Proprietà delle reti di regolazione genica

Come accennato in precedenza, quando parliamo di *Reti di Regolazione Genica* intendiamo tutte le interazione tra geni, proteine ed altri metaboliti,

che si susseguono durante la fase post-trascrizionale e che permettono ad una cellula di controllare le proprie funzioni interne ed esterne. Queste reti possono, inoltre, essere rappresentate come un grafo diretto in cui i nodi rappresentano i geni o le proteine da essi codificate, mentre gli archi orientati rappresentano l'azione regolatoria tra nodi diversi.
Una delle metodologie più usate per l'inferenza delle reti di regolazione genica è il *Reverse Engineering*, cioè l'insieme dei metodi, utili a ricostruire la complessa rete di regolazione e controllo dell'output dinamico del sistema osservato. Data la complessità del sistema analizzato, gli approcci di reverse engineering per lo studio di interazione tra geni e, proteine è in genere limitato a piccole parti della rete di regolazione.
Nel corso degli ultimi anni le reti di regolazione genica e metabolica sono state ampiamente studiate e analizzate, permettendo di definirne alcune proprietà fondamentali:

- **Scale-free:** la stragrande maggioranza dei geni non sono regolatori, e la distribuzione del grado d'uscita dei fattori di trascrizione (cioè la distribuzione del numero di nodi regolati da ciascun fattore di trascrizione) segue una legge esponenziale della forma

  $$P(k) = \alpha k^{-\gamma}$$

  dove k è il grado d'uscita dei fattori di trascrizione, γ è un valore compreso tra $2 < \gamma < 3$ e α è un fattore di normalizzazione[21]. Reti di regolazione che condividono questa proprietà sono note in letteratura come *scale-free networks*[11].

- **Sparsità:** è una proprietà connessa alla precedente, e riguarda l'esiguo numero di relazioni presenti in una rete di regolazione genica[9]. Per una rete di n geni, il numero totale di regolazioni è nell'ordine di $O(n)$, molto inferiore al numero di regolazioni dirette possibili, n^2.

- **Alto coefficiente di clustering:** il coefficiente di clustering di un nodo i è definito come il rapporto tra il numero di archi che collegano i nodi adiacenti ad i e il numero totale di possibili archi tra quest'ultimi[10]. In una rete "sparsa" casuale, il coefficiente di clustering medio tende ad avere un valore basso, che diminuisce all'aumentare dei nodi. In una rete di regolazione metabolica invece, il coefficiente di clustering ha un valore più elevato ed è indipendente dal numero di nodi che formano la rete.

- **Small-World:** il cammino metabolico tra coppie di geni tende ad essere più corto rispetto al valore atteso in una rete casuale. Questo tipo di reti sono note in letteratura come *small-world networks*[10].

- **Network Motifs:** alcuni modelli di regolazione, ad esempio uno schema di connessione tra tre o più geni, presentano un significativo valore

di apparizione nelle reti di regolazione[22], sono cioè molto frequenti. Si suppone che ogni schema di regolazione abbia una particolare funzione biologica e che venga conservato attraverso diverso tipi di organismi.

Tutte queste proprietà delle reti possono essere sfruttate nel processo di Reverse Engineering, sia come informazione *a priori* per gli algoritmi sia per identificare reti putative dedotte dagli stessi dati.

Capitolo 2

Stato dell'arte e dati sperimentali

Negli ultimi anni in letteratura sono stati proposti diversi metodi di Reverse Engineering per l'inferenza di Reti di Regolazione Genica (*Gene Regulatory Networks*) a partire da esperimenti di osservazione in stato stazionario di perturbazioni sistematiche dell'espressione dei geni di un organismo. Alcuni di essi si basano sull'osservazione di variabili fenotipiche, cioè sugli aspetti esteriori dell'organismo mutato, mentre altri si basano sull'osservazione degli effetti di una mutazione sull'espressione dei geni stessi.
GenePath[15, 16] è un sistema appartenente alla prima categoria: a partire da un'astrazione qualitativa dei dati è in grado di inferire le regolazioni della rete in esame e visualizzarle in un grafo in cui i nodi rappresentano i geni: le varibili fenotipiche, mentre gli archi orientati rappresentano l'azione regolatoria tra nodi diversi.
Un metodo complementare, basato sempre su un'astrazione qualitativa dei dati ma che utilizza informazioni su profili di espressione genica, è quello elaborato da F. Sambo[30].
Nel seguito verrà illustrato quest'ultimo modello, che rappresenta il framework di partenza per lo studio condotto in questa tesi. Successivamente verranno presentati gli stumenti applicativi utilizzati per lo sviluppo e la realizzazione dell'algoritmo. Infine, dopo aver descritto i dataset utilizzati per testare e validare i risulati ottenuti dall'algoritmo, si discuterà dei metodi di misurazione delle performance per questa tipologia di studio e delle difficoltà del problema di inferenziazione delle Reti di Regolazione Genica.

2.1 Framework di partenza

Lo studio condotto da F. Sambo[30] ha come obbiettivo l'inferenza di Reti di Regolazione Orientate, partendo da esperimenti di perturbazione sistemica dei singoli geni. L'algoritmo è stato testato su due insiemi di dati simulati,

che verranno descritti nella sezione 2.3.
L'analisi di questi dati segue un approccio qualitativo, nel quale ogni esperimento di knock-out viene confrontato con il caso base identificando le differenze tra le due osservazioni e cercando di identificare delle relazioni causali a partire dalla lista di differenze ottenuta. In figura 2.1 vengono mostrati graficamente i profili di espressione genica ottenuti da ogni esperimento di knock-out, a fianco dei quali viene mostrato l'output dell'esperimento di wild-type per la medesima rete composta da 20 geni.
Sapendo che ogni riga della matrice rappresenta un gene, mentre ogni co-

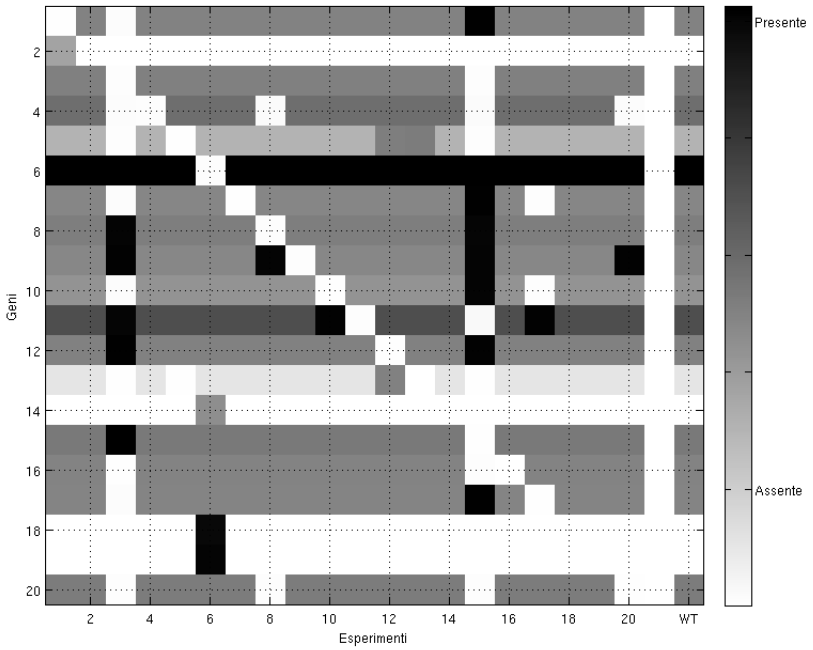


Fig. 2.1: Rappresentazione grafica dell'output di esperimenti di knock-out per una rete di 20 geni. A fianco il risultato di un'esperimento di wild-type. Ogni riga rappresenta un gene, mentre ogni colonna un esperimento. L'intensità è proporzionale al livello di espressione del gene, dall'assenza (bianco) al massimo livello misurato (nero).

lonna un esperimento, si può notare che la maggioranza degli esperimenti presenta un profilo di espressione genica che non differisce molto dal profilo osservato nel wild-type. Ciò è principalmente dovuto al fatto che le reti di regolazione genica considerate sono del tipo *scale-free*, dove cioè la stragrande maggioranza dei geni non sono regolatori.
In figura 2.2 viene mostrata l'informazione qualitativa ottenuta sottraendo da ogni esperimento di knock-out il wild-type, esprimibile in numero di differenze individuate. Come si può notare, nonostante le reti siano scale-free, vi è un ristretto numero di geni, il cui spegnimento forzato, modifica il livello

di espressione di molti geni.
Nella figura 2.1 l'intensità di un box è proporzionale al livello di espressio-

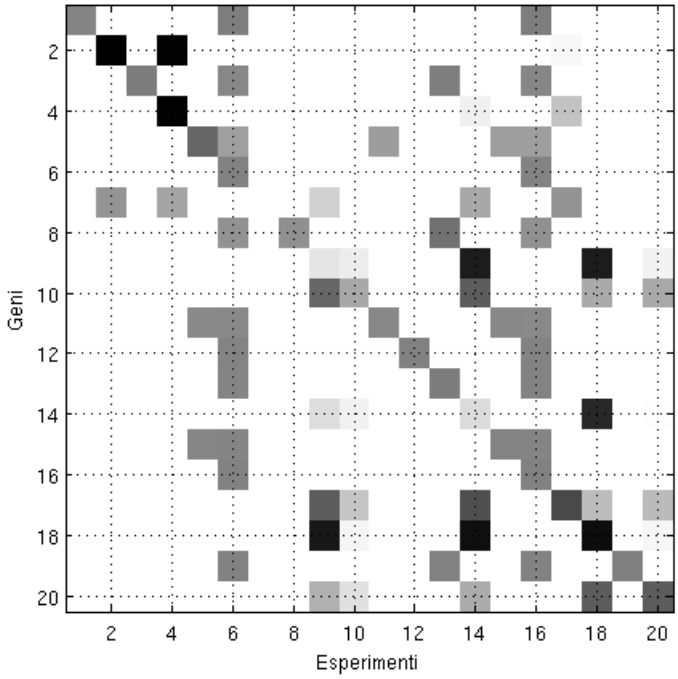


Fig. 2.2: Rappresentazione grafica dell'informazione qualitativa ottenuta dalla sottrazione del wild-type da ogni esperimento di knock-out, per la rete mostrata in precedenza.

ne normalizzato del gene, dall'assenza (bianco) al massimo valore misurato (nero), mentre in figura 2.2 l'intensità di un box rappresenta la variazione del livello del gene rispetto al caso di riferimento, da nessuna differenza misurabile (bianco) alla massima variazione misurata (nero).
Per ovviare a problemi dovuti a piccole variazioni tra il livello d'espressione osservato nell'esperimento di knock-out ed il wild-type, i valori delle variazioni inferiori ad una soglia θ vengono posti uguali a zero.
Ogni gene viene classificato come *osservabile* o *non osservabile*. Un gene è definito *osservabile* se la differenza tra il suo livello d'espressione nel wild-type e nell'esperimento di knock-out è maggiore della soglia θ, ciò implica che l'insieme dei geni presenti nella colonna della matrice degli effetti, corrispondente a quel preciso esperimento di knock-out, viene definito *lista degli effetti osservabili*. Contrariamente, alla seconda categoria appartengono tutti quei geni che hanno un livello d'espressione prossimo allo zero nel wild-type e per i quali un esperimento di knock-out non porta nessuna informazione aggiuntava ai fini dello studio condotto, ad esempio l'esperimento 7 in figura e da non confondere con gli esperimenti 1, 3, 8, 12 e 19 dove il

gene soppresso è osservabile, ma non ha nessuna effetto osservabile.
Considerando un generico gene osservabile x e definendo $eff(x)$ la lista dei suoi effetti osservabili, l'algoritmo sviluppato da F. Sambo è in grado di inferire regolazioni dirette tra due geni x, y, nella forma:

$$x \Rightarrow y$$

seguendo le seguenti regole:

- *Single Effect Rule*: nel caso in cui gli effetti osservati di x consistano in un solo gene y, allora è sempre possibile inferire la regolazione $x \Rightarrow y$;
- *Simple Inclusion Rule*: nel caso esistano due geni x, y tali per cui $eff(x) = \{y, eff(y)\}$, cioè quando y e tutti i suoi effetti appartengono alla lista degli effetti di x, si può inferire che $x \Rightarrow y$. Ciò è dovuto al fatto che la perturbazione causata dalla soppressione di x, coinvolge tutti e soli gli effetti osservabili di y;
- *Strict Inclusion Rule*: nel caso esistano due geni x, y tali per cui $eff(x) = \{y, eff(y), K\}$, cioè la lista degli effetti di x contiene y ed i suoi effetti, ma contiene anche un ulteriore insieme (K) di geni. Per poter asserire $x \Rightarrow y$ è necessario verificare che nessuno dei geni appartenenti a K si interponga tra x e y. Ciò è verificato se $\forall k \in K$ vale almeno una delle seguenti condizioni:
  - k è osservabile e y non è un effetto di k;
  - esiste z tale che k è un effetto di z, ma y non lo è.

Le regolazioni identificate permettono così di costruire una rete di regolazione orientata, senza però identificare il segno della regolazione stessa: positiva (attivazione) o negativa (inibizione).
Esistono inoltre dei casi particolari nei quali l'algoritmo non è in grado di inferire nessuna regolazione: si tratta di quei casi in cui alcuni geni si autoregolano tra di loro, formando cioè dei loop. L'algoritmo è in grado di identificare alcuni di questi casi, in particolare quelli in cui $\{x, eff(x)\} = \{y, eff(y)\}$, ma non considerando il segno delle regolazioni non è in grado di disambiguare questi casi particolari.
Per quanto riguarda la complessità computazionale, calcolata nel *caso peggiore*, l'algortimo si attesta su un valore di $O(n^4)$, con n il numero di geni costituenti la rete di regolazione che si vuole ricostruire.

2.2 Applicazioni

In questa sezione verranno descritte brevemente le applicazioni software utilizzate nel corso di questa tesi: Matlab[6], per quanto riguarda l'elaborazione dei dati e l'inferenza delle regole, GraphViz[1], per quanto riguarda la creazione dei grafi.

2.2.1 Matlab

MATLAB (abbreviazione di Matrix Laboratory)[6] è un ambiente per il calcolo numerico e l'analisi statistica che comprende anche l'omonimo linguaggio di programmazione creato dalla *The MathWorks*. MATLAB consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e interfacciarsi con altri programmi. Nonostante sia specializzato nel calcolo numerico, un toolbox opzionale interfaccia MATLAB con il motore di calcolo simbolico di Maple. MATLAB è usato da milioni di persone nell'industria, nelle università e funziona su diversi sistemi operativi, tra cui Windows, Mac OS, GNU/Linux e Unix.
La scelta di utilizzare Matlab come strumento/linguaggio per lo sviluppo di questa tesi è dovuta alla struttura matriciale dei dati da elaborare, ottenuti da esperimenti di DNA-microarray e in secondo luogo al modello di partenza, sviluppato anch'esso utilizzando quest'ambiente.

2.2.2 GraphViz

GraphViz (abbreviazione di *Graph Visualization Software*)[1] è un programma open source avviato da AT&T Research Labs per disegnare grafi descritti nel linguaggio DOT, distribuito con licenza Common Public License.
GraphViz non ha un'interfaccia grafica ma dev'essere invocato da riga di comando. Per permettere l'interfacciamento con Matlab ci si è avvalsi della libreria gratuita GraphViz2Matlab [19], opportunamente modificata. É stato infatti inizialmente necessario risolvere alcuni bug presentati dalla libreria e successivamente apportare alcune modifiche per renderla adatta allo scopo della tesi. In particolare é stata modificata la modalità di identificazione dei nodi, aggiungendo la possibilità di settare un nome ai nodi. Infine è stato introdotto un nuovo meccanismo per la creazione degli archi che permettesse di colorarli in base al tipo di regolazione ed evitasse la sovrapposizione degli stessi per le regolazioni reciproche tra due nodi (loop).

2.3 Dati sperimentali

In letteratura, il numero di reti di regolazione aventi un alto grado di certezza e confidenza sono in numero molto esiguo. Per questo motivo e per l'impossibilità di definire degli standard di riferimento[27, 26], i test e la validazione dei risultati ottenuti con l'algoritmo sviluppato in questa tesi sono stati eseguiti con dati ottenuti con l'ausilio di simulatori di reti biologiche o ricandoli da reti di regolazione di organismi biologici noti in letteratura.
Le osservazioni dei profili espressione genica possono essere assimilate a osservazioni di tipo *steady state* su due tipologie di esperimenti DNA-microarray: senza nessuna perturbazione esterna (*wild-type*) e con una perturbazione atta a inibire la trascrizione di un singolo gene (*knock-out*). L'esperimento di

knock-out è stato condotto sistemicamente per ogni gene appartenente alla rete e per ogni taglia della rete.
I dataset utilizzati si possono inoltre, dividere sostanzialmente in due categorie: *in assenza rumore* e *con rumore*.

2.3.1 Dataset in assenza di rumore

La prima tipologia di dataset impiegato per il testing e la validazione sia del framework di partenza, sia dell'algoritmo che si è sviluppato nello svolgimento di questo studio, è stata ottenuta grazie all'ausilisio del simulatore di reti biologiche *NetSim*[24].
Le reti costruite grazie a NetSim seguono un modello ipotizzato in [14] e rispettano alcune importanti caratteristiche di una vera rete di regolazione genica: distribuzione del grado d'uscita dei fattori di trascrizione *scale-free*, *alto coefficiente di clustering* (indipendente dal numero di nodi della rete) e lunghezza dei cammini metabolici più breve rispetto al valore atteso (*small-world*).
Per avere un set di dati che permettesse di coprire più livelli di complessità, sono state generate un insieme di reti con diverso numero di geni, cioè 10, 20, 50 e 100. Inoltre, per ogni taglia sono presenti 20 diverse reti, per permettere una migliore valutazione del comportamento medio dell'algoritmo nei diversi casi.

2.3.2 Dataset con rumore

Il progetto DREAM (Dialogue for Reverse Engineering Assessments and Methods [28, 29]) ha come obiettivo principale lo sviluppo di una una maggiore interazione tra ciò che riguarda la teoria e ciò che riguarda gli esperimenti condotti in merito alle reti di regolazione cellulare, cercando di ottenere un confronto equo dei punti di forza e di debolezza dei metodi di Reverse Engineering e un chiaro senso di affidabilità dei modelli di rete che producono. Con questo scopo, ogni anno vengono pubblicate nuove sfide di Reverse Engineering, per permettere ai ricercatori di competere sulle loro abilità di dedurre le reti di regolazione partendo da dati simulati.
Abbiamo sfruttato uno dei dataset dell'edizione più recente, DREAM4 In Silico Network Challenge del 2009, per valutare le prestazioni dell'algoritmo sviluppato. Il set di dati è costituito da una serie di cinque simulazioni diverse, su reti composte da 10 e 100 geni. La struttura di queste reti è stata ricavata da reti di organismi biologici ben noti in letteratura, vale a dire il *Saccharomyces Cerevisiae* e l'*Escherichia Coli*.
I profili d'espressione sono stati generati secondo un modello basato su equazioni differenziali stocastiche; ai profili generati è stato in seguito aggiunto del rumore, per simulare il rumore intrinseco negli esperimenti reali di DNA-microarray.

2.4 Misurazione della 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 *Precisione* (P) e il *Richiamo* (R)[12] e sono definite come:

$$P = \frac{tp}{tp + tf}$$

$$P = \frac{tp}{tp + fn}$$

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 presenti nella rete reale ma non identificate. Entrambe le misure variano nell'intervallo [0, 1] e possono essere utilizzate sia per i grafici orientati sia per i non orientati. Un riscontro grafico per illustrare ulteriormente le due misure è dato in Figura 2.3.

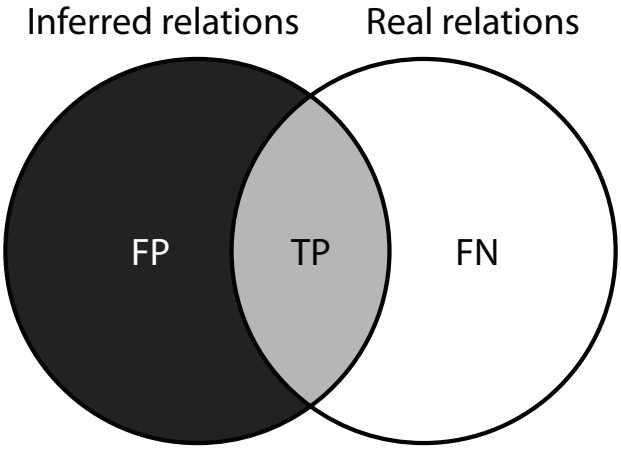


Fig. 2.3: Grafico esplicativo su Precisone e Richiamo. La circonferenza di sinistra mostra l'insieme delle relazioni individuate dall'algoritmo, mentre la circonferenza di destra mostra l'insieme delle relazioni reali della rete. Si può identificare la Precisione come il rapporto tra l'area dell'intersezione e l'area della circonferenza di sinistra $tp/(tp+fp)$, mentre il Richiamo come il rapporto tra l'area dell'intersezione e l'area della circonferenza di destra $(tp/(tp + fn)$.

Quando possibile, il confronto tra due algoritmi dovrebbe essere basato sulle misure di prestazione calcolate su una serie di problemi di Reverse Enginee-

ring, piuttosto che su un singolo caso. Si può quindi confrontare il comportamento medio degli algoritmi, usando ad esempio strumenti statistici come il test di Wilcoxon tra due campioni, e considerando come significative le differenze il cui p-value è inferiore ad una certa soglia[27].

2.5 Difficoltà e limitazioni del problema

La ricostruzione delle Reti di Regolazione Genica attraverso uno studio di Reverse Engineering è un problema di per sè difficile e lo stato dell'arte degli algoritmi presenti in letteratura è ancora lontano dal raggiungere buoni risultati su scala genomica ($O(10^3)$ geni). Una prima difficoltà risiede nel punto di osservazione: con le misurazioni con DNA-microarray si osserva la quantità di RNA tradotto in proteina in un determinato istante temporale, ma, come è noto, il maggior numero di interazioni biologiche avvengono a livello proteico, quindi dobbiamo fare affidamento su un'osservazione indiretta del fenomeno che si sta studiando[20]. Inoltre, non abbiamo la certezza di conoscere tutti i componenti che interagiscono durante questo processo di regolazione, ad esempio recenti studi hanno dimostrato che molecole di micro-RNA (μRNA [13]), cioè piccole molecole di RNA non codificanti, intervengono in realtà nel processo di regolazione, ma vengono escluse dagli esperimenti con microarray appunto per la loro caratteristica di essere non codificanti.
Per tutte queste ragioni, come avevamo accennato in precedenza, si osserva sempre una certa quantità di *rumore intrinseco* negli esperimenti di DNA-microarray, anche tra repliche della stessa osservazione. Un altro problema a cui far fronte, noto in letteratura come *maledizione della dimensionalità*[31]. Il problema generalmente si riferisce alle difficoltà incontrate quando si desidera adattare modelli a spazi a molte dimensioni. Mano a mano che il numero delle variabil in input (cioè, del numero di predittori) aumenta, diventa esponenzialmente più difficile trovare i punti di ottimo globale per i modelli. Quindi, risulta praticamente necessario eseguire un prescreening ed una preselezione su un grande insieme di variabili di input (predittori) che potrebbero essere di una qualche utilità per la previsione dei risultati d'interesse. Nel caso in esame ciò è dovuto alla disparità tra il numero di variabili osservate attraverso un esperimento di DNA-microarray ($O(10^3)$) e il numero usuale di osservazioni per ogni variabile ($O(10 \sim 10^2)$). Inoltre, anche nel caso in cui ci siano più osservazioni che variabili, esiste la possibilità che non tutti gli stati dinamici del sistema siano eccitati, ciò rende impossibile individuare alcune relazioni, semplicemente perchè non sono osservabili in quel determinato esperimento.
Questi sono probabilmente i motivi per cui gli algoritmi presenti in letteratura non sono in grado di ricostruire completamente le reti di regolazione, anche con dati simulati, grandi quantità di campioni e nessun rumore[23, 26].

Capitolo 3

Estensione del Framework di Partenza

L'algoritmo sviluppato nel corso di questa tesi ha come obbiettivo l'identificazione di possibili relazioni causa-effetto tra coppie di geni, messe in luce da un confronto qualitativo tra i profili d'espressione dei geni stessi.
In questo capitolo verrano mostrate le estensioni al framework di partenza, che hanno permesso l'identificazione della tipologia di regolazione, positiva o negativa, e la capacità di inferire un maggior numero di regolazioni. Dopo aver discusso di alcuni casi particolari, nei quali alcuni geni si autoregolano formando dei loop, si condurrà un'analisi sull'algoritmo prodotto. Infine verrà mostrato un confronto tra la rete inferita in un'analisi condotta dell'algoritmo e la rete reale.

3.1 Gestione del livello d'espressione

Uno degli obbiettivi principali dello studio condotto è stata l'introduzione della gestione del segno delle regolazioni nel framework di partenza, presentato nella sezione 2.1. Come indicato in precedenza, l'incapacità del framework di base di inferire il segno delle regolazioni, oltre a impedire la disambiguazione dei casi di autoregolazione tra geni, forniva un'informazione incompleta sull'interazione tra i geni stessi.
L'informazione qualitativa è ottenuta, come nel framework di partenza, dalla sottrazione del profilo di espressione genica dell'esperimento wild-type da ogni esperimento di knock-out, considerando soltanto i geni che presentano una differenza assoluta maggiore di una soglia θ.
In figura 3.1 viene mostrata la nuova rappresentazione grafica dell'informazione qualitativa. Come si può notare, a differenza della rappresentazione in figura 2.2 nella quale venivano mostrati quali geni avessero un livello d'espressione differente dal wild-type, nella rappresentazione attuale non solo viene mostrata questa caratteristica, ma viene inoltre indicato in che di-

rezione il livello d'espressione del gene vari rispetto al caso di riferimento: positiva (sovra-espressione) o negativa (sotto-espressione).

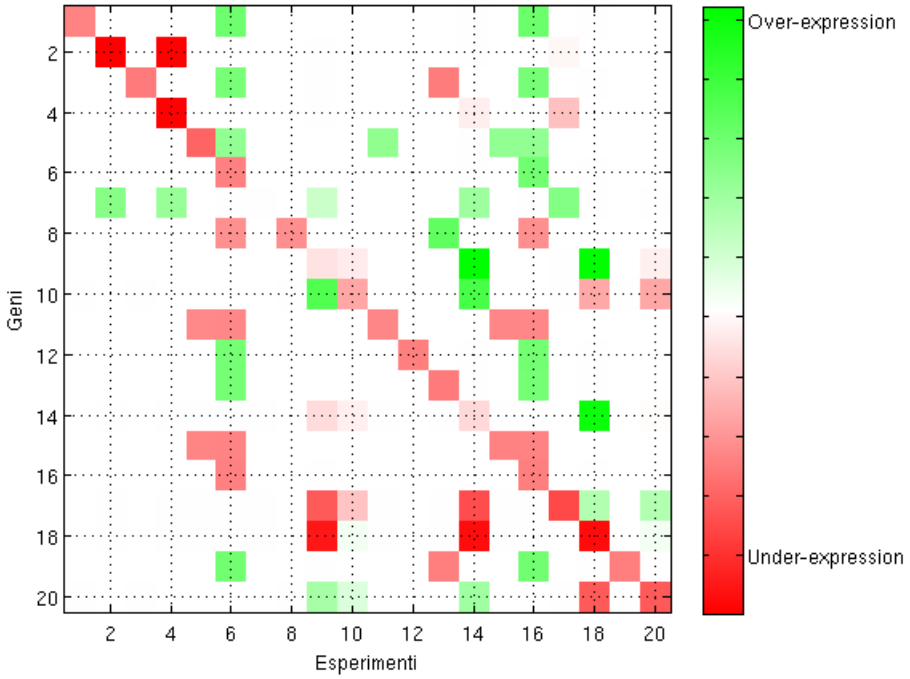


Fig. 3.1: Rappresentazione grafica dell'informazione qualitativa ottenuta dalla sottrazione del wild-type da ogni esperimento di knock-out, per la rete mostrata in sezione 2.1. L'intensità è proporzionale al livello di espressione del gene in relazione al wild-type, dal massimo livello di sotto-espressione misurato (rosso), al massimo livello di sovre-espressione misurato (verde), passando per un'eguale livello (bianco).

Come nella rappresentazione originale (fig. 2.2), anche in questo caso ogni riga della matrice rappresenta un gene, mentre ogni colonna la lista degli effetti osservabili in un esperimento. Analogamente vengono definiti come *non osservabili* quei geni per i quali l'elemento presente sulla diagonale della matrice degli effetti abbia una differenza assoluta tra il livello d'espressione osservato nell'esperimento di wild-type e nell'esperimento di knock-out minore della soglia θ. Considerando la rete di 20 geni già mostrata in precedenza, una rappresentazione testuale degli effetti osservati per ogni gene è mostrata nella tabella 3.1. Considerando un generico gene osservabile x e definendo $eff(x)$ la lista degli effetti osservati durante l'esperimento di knock-out sul gene stesso, come nel framework di partenza, anche il metodo attuale è in grado di inferire regolazioni dirette tra due geni x, y, identificando nel caso attuale il segno della regolazione:

$$x \Rightarrow \pm y$$

-7	⇒	non osservabile									
-1	⇒										
-3	⇒										
-8	⇒										
-12	⇒										
-19	⇒										
-2	⇒	7									
-11	⇒	5									
-4	⇒	-2	7								
-5	⇒	-11	-15								
-15	⇒	5	-11								
-13	⇒	-3	8	19							
-17	⇒	-2	-4	7							
-10	⇒	-9	-14	-17	18	20					
-18	⇒	9	-10	14	17	-20					
-20	⇒	-9	-10	-14	17	18					
-9	⇒	7	10	-14	-17	-18	20				
-14	⇒	-4	7	9	10	-17	-18	20			
-6	⇒	1	3	5	-8	-11	12	13	-15	-16	19
-16	⇒	1	3	5	6	-8	-11	12	13	-15	19

Tab. 3.1: Rappresentazione testuale degli effetti osservabili per ogni gene.

dove x rappresenta il gene regolatore, mentre y il gene regolato. La regolazione, a differenza del framework di partenza, può essere positiva o negativa in base al segno di y.
Come si può notare, nella tabella 3.1 i geni regolatori hanno tutti segno negativo, ciò è dovuto al fatto che gli esperimenti considerati sono di knock-out, cioè di soppressione "forzata" di un gene, identificabile idealmente come una regolazione negativa sul gene a monte dell'esperimento. Questo aspetto implica, inoltre, che le regolazioni identificate dall'algoritmo hanno tutte segno complementare a quello mostrato nella tabella 3.1. Ad esempio la regolazione inferita non è $-2 \Rightarrow 7$, bensì $2 \Rightarrow -7$.

3.2 Inferenza delle regolazioni

L'informazione aggiuntiva, estrapolata dalla sottrazione del wild-type da ogni esperimento di knock-out, sulla variazione del livello di espressione di un gene ha permesso di asserire un'indicazione più precisa sulle regolazioni tra coppie di geni ma, al contempo, ha richiesto un'adattamento delle regole di inferenza formulate da F. Sambo e mostrate in precedenza nel capitolo 2.1.
Considerando anche in questo caso un generico gene osservabile x e definendo $eff(x)$ la lista degli effetti osservati durante l'esperimento di knock-out sul gene stesso, le nuove formulazioni sono:

- *Single Effect Rule*: nel caso in cui gli effetti osservati di x consistano in un solo gene $\pm y$, allora è sempre possibile inferire la regolazione $x \Rightarrow \mp y$. Due esempi in tabella 3.1, sono $2 \Rightarrow -7$ e $11 \Rightarrow -5$, , visibili anche in figura 3.3.

- *Simple Inclusion Rule*: nel caso esistano due geni x, y tali che $eff(x) = \{y, eff(y)\}$, cioè quando y e tutti i suoi effetti appartengono alla lista degli effetti di x, è possibile inferire $x \Rightarrow y$. Analogamente, nel caso opposto in cui $eff(x) = \{-y, -eff(y)\}$, cioè quando gli effetti di x sono l'unione di $-y$ e dell'insieme degli effetti di y cambiati di segno, è possible inferire $x \Rightarrow -y$. L'asserzione di questa tipologia di regolazione è possibile in quanto la perturbazione causata dalla soppressione del gene x si manifesta in maniera congrua su y e su tutti gli effetti di y. Due esempi in tabella 3.1 sono $15 \Rightarrow 11$ e $17 \Rightarrow 4$, visibili anche in figura 3.3.

- *Strict Inclusion Rule*: nel caso esistano due geni x, y tali che $eff(x) = \{y, eff(y), K\}$, cioè la lista degli effetti di x contiene y e $eff(y)$, ma contiene anche un ulteriore insieme (K) di geni. Come nella regola precedente, esiste un caso opposto, nel quale $eff(x) = \{-y, -eff(y), K\}$. Per poter asserire $x \Rightarrow \mp y$ è necessario verificare che nessuno dei geni appartenenti a K si interponga tra x e y. Ciò è verificato se $\forall k \in K$ vale, indipendentemente dal segno di y, almeno una delle seguenti condizioni:

  - k è osservabile e y non è un effetto di k;
  - esiste z tale che k è un effetto di z, ma y non lo è.

Successivamente, sono state definite quattro ulteriori regole di inferenza ed una nuova struttura dati contenente la lista dei geni che direttamente e/o indirettamente regolano un gene, ottenuta leggendo la matrice degli effetti non più per colonna, bensì per riga. Definendo $per(x)$ la lista dei geni perturbatori di un generico x, cioè di quei geni che ne determinano una variazione osservabile del livello d'espressione, riportiamo in tabella 3.2 una rappresentazione testuale della struttura per la rete composta da 20 geni considerata in precedenza.
Mantenendo le considerazioni fatte in precedenza, cioè prendendo un generico gene x e definendo $eff(x)$ la lista dei suoi effetti osservabili e $per(x)$ la lista dei suoi geni perturbatori, si possono definire le nuove regole come:

- *Inverse Rule*: nel caso esistano due geni x, y tali per cui $y \in eff(x)$ e $per(y) = \{x\}$, cioè y è un effetto osservabile solamente nell'esperimento di knock-out di x, allora $x \Rightarrow y$. Si può dire che questa regola sia complementare alla regola dell'*effetto singolo*, in quanto invece di leggere la matrice per colonne ed individuare i geni con un solo effetto

1	$\Leftarrow$	6	16			
2	$\Leftarrow$	4	17			
3	$\Leftarrow$	6	13	16		
4	$\Leftarrow$	14	17			
5	$\Leftarrow$	6	11	15	16	
6	$\Leftarrow$	16				
7	$\Leftarrow$	2	4	9	14	17
8	$\Leftarrow$	6	13	16		
9	$\Leftarrow$	10	14	18	20	
10	$\Leftarrow$	9	14	18	20	
11	$\Leftarrow$	5	6	15	16	
12	$\Leftarrow$	6	16			
13	$\Leftarrow$	6	16			
14	$\Leftarrow$	9	10	18	20	
15	$\Leftarrow$	5	6	16		
16	$\Leftarrow$	6				
17	$\Leftarrow$	9	10	14	18	20
18	$\Leftarrow$	9	10	14	20	
19	$\Leftarrow$	6	13	16		
20	$\Leftarrow$	9	10	14	18	

Tab. 3.2: Rappresentazione testuale dei geni perturbatori per ogni gene.

osservabile, la matrice viene letta per righe, cercando i geni regolati solamente da un altro gene.

- *Last Parent Rule*: nel caso in cui $\forall k \in eff(x) : length(eff(k)) = 0$, k osservabile e $per(k) \in per(x)$, cioè tutti gli effetti osservabili di x siano a loro volta osservabili, non abbiano nessun effetto osservabile e la lista dei geni perturbatori per ognuno di essi sia sottoinsieme della lista dei perturbatori di x, vale la regola $x \Rightarrow k$. Con riferimento alla tabella 3.1, un esempio delle regolazioni individuate sono: $13 \Rightarrow 3$, $13 \Rightarrow -8$ e $13 \Rightarrow -19$, osservabile anche in figura 3.3.

- *Ugly Duckling Rule*: nel caso esistano due geni x, y, con y gene non osservabile e tali per cui $eff(x) = \{y, K\}$, $\forall k \in K$: k osservabile e y non è un effetto osservabile di k, allora $x \Rightarrow y$. Sapendo y non è regolato direttamente da nessuno dei rimanenti k geni appartenenti a $eff(x)$, si può desumere che y sia per forza di cose un effetto di x, nulla però si può affermare per gli altri geni, in quanto, essendo per ipotesi iniziale y non osservabile non è possibile stabilire se si interponga tra x ed essi.

- *Alternative Pathway Rule*: nel caso esistano due geni x, y tali per cui $eff(x) = \{y, -eff(y)\}$ o $eff(x) = \{-y, eff(y)\}$, cioè quando la perturbazione causata dalla soppressione del gene x si manifesta in maniera opposta su y e su tutti gli effetti di y, con y non appartenente

a nessun anello di autoregolazione e con tutti i suoi effetti osservabili, allora $\forall k \in eff(x) : x \Rightarrow k$. Il fatto che i segni degli effetti di x abbiamo segno opposto a quello degli effetti di y, implica che esiste un cammino alternativo tra x ed il generico effetto k, oltre a quello che prevede il passaggio attraverso il gene y, come mostrato in figura 3.2; in questo caso x corrisponde al gene 6 e y al gene 5. La lista degli effetti osservabili del knock-out di x è difatti $\{-5, 2, -9\}$, mentre quella di y è $\{-2, 9\}$.

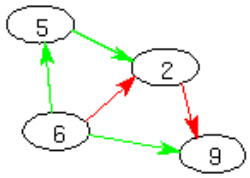


Fig. 3.2: Rappresentazione grafica di alcune regole inferite grazie all'Alternative Pathway Rule. In questa porzione di rete composta sempre da 20 geni, ma differente da quella considerata in precedenza, il gene 6 regola direttamente i geni 5,2 ed al contempo indirettamente i geni 2 e 9, in quanto regolati da 5.

Grazie a queste nuove regole è possibile inferire un numero maggiore di regolazioni, ma soprattutto grazie alle regole *Alternative Pathway* e *Last Parent* è possibile individuare più di una sola regola a partire da una singola riga di effetti; nel framework di partenza questo non era consentito.
Come si vedrà nella sezione 3.5, l'algoritmo associa ad ogni regolazione inferita un differente peso in base alla regola che ne ha permesso l'asserzione. È così possibile identificare sempre l'insieme delle regolazioni inferite grazie ad una regola e stimare la precisione ed il richiamo per ciascuno di essi, oppure limitare la ricerca delle regolazioni ad un sottoinsieme delle regole esposte. Come è facile immaginare, un peso positivo identifica una regolazione positiva, mentre un peso negativo identifica una regolazione negativa.

3.3 Rappresentazione Grafica delle reti inferite

Una rete di regolazione genica può essere rappresentata come un grafo diretto in cui i nodi rappresentano i geni o le proteine da essi codificate, mentre gli archi orientati rappresentano l'azione regolatoria tra nodi diversi. Sfruttando questa caratteristica è stata introdotta la possibilità di rappresentare le regolazioni inferite come dei grafi nei quali gli archi possono assumere due colorazioni diverse in base al tipo di regolazione che rappresentano: positiva (verde) o negativa(rossa).
In figura 3.3 vengono mostrate le regolazioni inferite nella rete da 20 geni considerata in precedenza e le cui liste di effetti osservabili per ogni gene

sono mostrate in tabella 3.1.

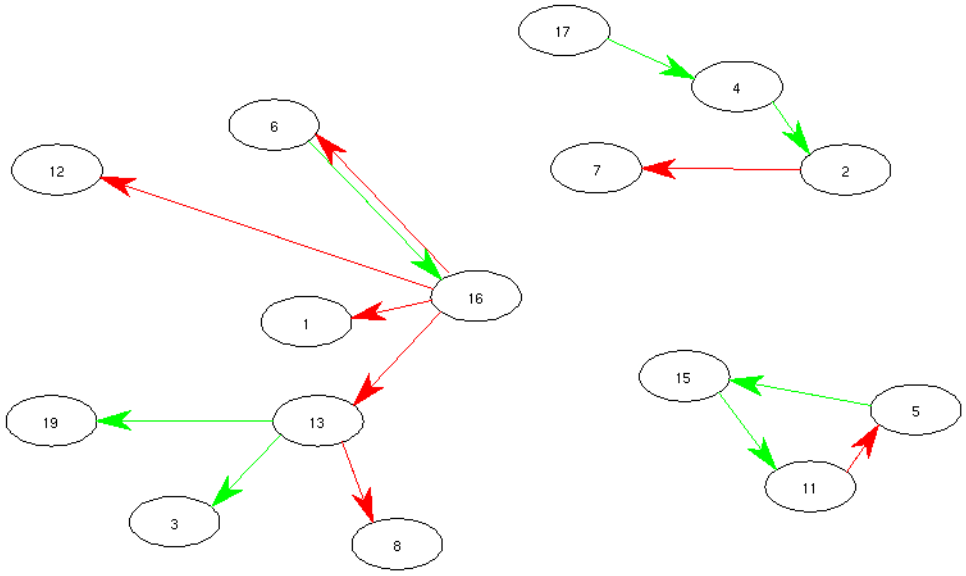


Fig. 3.3: Rappresentazione grafica della Rete di Regolazione Genica inferita, dove i nodi rappresentano i geni, mentre gli archi orientati rappresentano l'azione regolatoria tra i nodi: positiva (verde) o negativa (rossa).

Per la creazione dei grafici è stato utilizzato uno strumento esterno: GraphViz[1]. Per permettere l'interfacciamento con Matlab ci si è avvalsi della libreria gratuita GraphViz2Matlab [19], opportunamente modificata. É stato infatti inizialmente necessario risolvere alcuni bug presentati dalla libreria e successivamente apportare alcune modifiche per renderla adatta allo scopo. In particolare é stata modificata la modalità di identificazione dei nodi, aggiungendo la possibilità di settare un nome ai nodi. Infine è stato introdotto un nuovo meccanismo per la creazione degli archi che permettesse di colorarli in base al tipo di regolazione ed evitasse la sovrapposizione degli stessi per le regolazioni reciproche tra due nodi (loop). Come si vedrà nella prossima sezione, la rappresentazione grafica delle reti inferite è stata utilizzata anche per rappresentare gli anelli di autoregolazione che l'algoritmo non è in grado di disambiguare in maniera autonoma, come mostrato in figura 3.5(b).

3.4 Autoregolazione: casi particolari

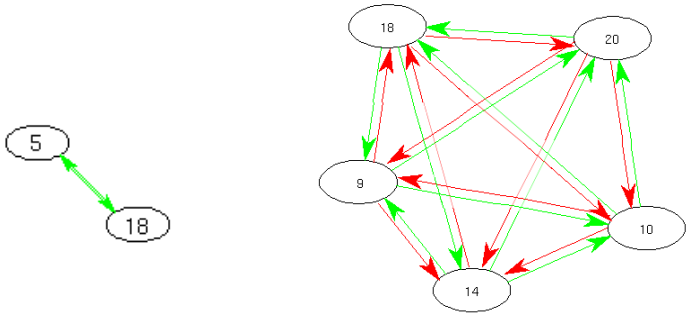
Durante l'esposizione del framework di partenza (Sezione 2.1) è stata messa in luce l'impossibilità dell'algoritmo di disambiguare i casi in cui alcuni geni si trovino in autoregolazione reciproca, formando quindi dei loop. Nonostante l'algoritmo sia in grado di identificare correttamente i loop in cui

$\{x, eff(x)\} = \{y, eff(y)\}$, non considerando il segno delle regolazioni non è in grado di disambiguare questi casi particolari.
Sfruttando la gestione del segno delle regolazioni attuata in questo studio, è stato possibile analizzare e creare dei metodi in grado di disambiguare in maniera automatica alcuni casi di autoregolazione. In figura 3.4 sono mostrate le due tipologie di loop gestite dall'algoritmo, cioè i loop composti da 2 e 3 geni, indicando, inoltre, tutte le possibili regolazioni che possono esistere tra le coppie di geni appartenenti al loop.

(a) Loop con 2 geni. (b) Loop con 3 geni.

Fig. 3.4: Rappresentazione grafica di due tipologie di anello di autoregolazione.

L'identificazione dei loop è stata migliorata, sfruttando le due strutture dati presentate in precedenza: la lista degli effetti osservabili per un gene e la lista dei perturbatori di un gene. Considerando un generico gene osservabile x, se il risultato dell'intersezione (K) tra $eff(x)$ e $per(x)$ è diverso dall'insieme vuoto, significa che x si trova in autoregolazione con i geni appartenenti a K. Con questo metodo è possibile identificare loop che nel framework di partenza non venivano rilevati.
Se il numero dei geni coinvolti in un loop è maggiore di 3, l'algoritmo non è in grado di fornire autonomamente una risoluzione dell'anello, si è quindi scelto di sfruttare il metodo di creazione dei grafi mostrato in precedenza, per produrre in output un'immagine dei geni coinvolti nel loop e delle possibili regolazioni tra di essi. Un esempio di loop rilevato, ma non disambiguato autonomamente per il numero eccessivo di geni appartenenti all'anello di autoregolazione, è quello in figura 3.5(b) e formato dai geni 9, 10, 14, 18 e 20 della rete composta da 20 geni considerata in precedenza. Come si può notare in tabella 3.1, le liste degli effetti osservabili associate a questi geni hanno dimensione differente e non sarebbero state identificate dall'algoritmo di partenza.
In figura 3.5(a) viene mostrato un'ulteriore occasione in cui l'algoritmo non è in grado di disambiguare un loop. Si tratta del caso in cui una coppia di geni si trovi in autoregolazione reciproca e i segni delle due regolazioni siano concordi, o entrambi positivi o entrambi negativi.
Per quanto riguarda le modalità di risoluzione delle due tipologie di loop considerati e mostrati in figura 3.4, si basa sui segni osservati dei geni in autoregolazione per quanto riguarda i loop composti da 3 geni, mentre il procedimento è più complesso per i loop da 2 geni, per i quali viene pri-



(a) Regolazioni con segno concorde.

(b) Numero di geni in autoregolazione troppo elevato.

Fig. 3.5: Rappresentazione grafica di due tipologie loop che l'algoritmo non è in grado di disambiguare in maniera autonoma.

ma verificato se esistano relazioni di *simple inclusion* o *strict inclusion* tra i due geni e in seguito viene effettuata un'operazione di disambiguazione basandosi sui segni dei geni. Se il primo controllo fornisce esito positivo, viene attivato un meccanismo di *backward analysis*, nel quale vengono ricercate possibili regolazioni non inferite in precedenza per l'ambiguità del loop. Un esempio di loop a due geni nella rete da 20 geni citata in precedenza e correttamente risolto è tra i geni 6 e 16. In questo caso, il meccanismo di *backward analysis* è in grado di inferire 3 ulteriori regolazioni, in particolare: $16 \Rightarrow -13$, $16 \Rightarrow -1$ e $16 \Rightarrow -12$, come mostrato nella figura 3.3.

3.5 Algoritmo

Le estensioni applicate al framework di partenza e mostrate in precedenza hanno richiesto una riscrittura totale dell'algoritmo di *reverse engineering*, basato sul paradigma del Ragionamento Qualitativo, proposto da F. Sambo in [30]. Il listato 3.1 mostra lo pseudo-codice del nuovo core, che permette, oltre all'inferenza di regolazioni con segno, l'individuazione di tutti gli anelli di autoregolazione tra i geni ed una rappresentazione a grafo della rete di regolazione inferita. Nella sezione precedente è stata discussa la modalità di risoluzione autonoma per i loop individuati durante l'analisi. Dal punto di vista implementativo, la natura pseudo-ricorsiva di questo metodo ha richiesto la suddivisione dell'algoritmo in blocchi modulari. I listati 3.2 e 3.3 mostrano lo pseudo-codice dei due principali moduli utilizzati per lo scopo appena discusso.
Il nuovo algoritmo si basa anch'esso sul paradigma del Ragionamento Qualitativo, necessita quindi in *input* della matrice $\mathbf{A}^{n\times n}$ con i profili d'espressione genica ottenuti durante gli esperimenti di *gene knock-out* e del vettore $\mathbf{w}^{n\times 1}$ con il profilo d'espressione genica osservato nell'esperimento di *wild-type*. È

AnalyzeK0($\mathbf{A}^{n\times n}$, $\mathbf{w}^{n\times 1}$, **Parametri Opzionali**)

1. Analisi dei **Parametri Opzionali** e inizializzazione variabili
2. Sottrai $\mathbf{w}$ da ogni colonna di $\mathbf{A}$ e salva i risultati nelle matrici $\mathbf{U}^{n\times n}$ e $\mathbf{D}^{n\times n}$ in base ai segni
3. Per ogni elemento in $\mathbf{D}$, se $\mathbf{D}[i,j] > -\theta$ allora $\mathbf{D}[i,j] \leftarrow 0$
4. Per ogni elemento in $\mathbf{U}$, se $\mathbf{U}[i,j] < \theta$ allora $\mathbf{U}[i,j] \leftarrow 0$
5. Somma le matrici $\mathbf{D}$ e $\mathbf{U}$ in $\mathbf{R}$
6. $\forall\, x$, $eff(x) \leftarrow$ indici degli elementi $\neq 0$ dalla x-esima colonna di $\mathbf{R}$
7. $\forall\, x$, $per(x) \leftarrow$ indici degli elementi $\neq 0$ dalla x-esima riga di $\mathbf{R}$
8. $n \leftarrow ncols(\mathbf{R})$
9. ▷ Prima fase di ricerca
10. **for** $i \leftarrow 1$ **to** $\underset{x}{\text{argmax}}\ length(eff(x))$ **do**
11. **for all** $x \mid length(eff(x)) = i$ **do**
12. **if** $i = 1$ **then**
13. ▷ Single Effect Rule
14. $C[eff(x), x] \leftarrow 1$
15. **else**
16. ▷ Verifica Presenza Loop
17. **if** $(K = eff(x) \cap per(x)) \neq \emptyset$ **then**
18. ResolveLoop(K)
19. ▷ Last Parent Rule
20. **else if** $\forall\, y \in eff(x)$:
21. $length(eff(y)) = 0$ **and** $per(y) \subset per(x)$ **then**
22. $C[y, x] \leftarrow 1$
23. ▷ Ugly Duckling Rule
24. **else if** $\exists\ y \in eff(x) \mid y$ non osservabile **and**
25. $(per(y) \cap eff(x)) = \emptyset$ **then**
26. $C[y, x] \leftarrow 1$
27. ▷ Simple Inclusion Rule
28. **if** $\exists\ y \mid eff(x) = \{y, eff(y)\}$ **or** $eff(x) = \{-y, -eff(y)\}$ **then**
29. $C[y, x] \leftarrow 1$
30. ▷ Strict Inclusion Rule
31. **else if** $\exists\ y \mid eff(x) = \{y, eff(y), K\}$ **or**
32. $eff(x) = \{-y, -eff(y), K\}$ **then**
33. **if** $\forall\ k \in K$: (k osservabile **and** $x \notin eff(k)$) **or**
34. $(\exists\, z \mid k \in eff(z)$ **and** $x \notin eff(z))$ **then**
35. $C[y, x] \leftarrow 1$
36. ▷ Inverse Rule
37. **for all** $y \mid per(y) = x$ **and** $\forall\, k \in eff(x) : k$ osservabile **do**
38. $C[y, x] \leftarrow 1$

```
39. ▷ Seconda fase di ricerca
40. for i ← 1 to argmax_x length(eff(x)) do
41.     for all x | length(eff(x)) = i do
42.         ▷ Alternative Pathway Rule
43.         if ∃ y | eff(x) = {−y, eff(y)} or
44.                   eff(x) = {y, −eff(y)} then
45.             for all k ∈ eff(x) do
46.                 C[k, x] ← 1
47.         else if ∃ y | eff(x) = {y, eff(y), K} or
48.                       eff(x) = {−y, −eff(y), K} then
49.             ▷ Last Parent Rule
50.             if ∀ k ∈ K : k osservabile and length(eff(k)) = 0 and
51.                           per(k) ⊂ per(x) then
52.                 C[k, x] ← 1
53.                 ▷ Alternative Pathway Rule
54.             else if ∀ k ∈ {y, eff(y)} : k non appartiene ad un loop then
55.                 C[k, x] ← 1
56. return C
```

Lis. 3.1: Pseudo codice del core dell'algoritmo.

```
    ResolveLoop(K)
1.  if length(k) = 2 then
2.      x = K(1); y = K(2)
3.      if SimpleInclusion(x,y) or StrictInclusion(x,y) then
4.          C[x, y] ← 1
5.          BackwardAnalysis(x,y)
6.      else if SimpleInclusion(y,x) or StrictInclusion(y,x) then
7.          C[y, x] ← 1
8.          BackwardAnalysis(y,x)
9.      s = GetRegulation(K)
10.     for all (x, y) ∈ s do
11.         C[y, x] ← 1
12. else if length(K) = 3 then
13.     s = GetRegulation(K)
14.     for all (x, y) ∈ s do
15.         C[y, x] ← 1
16. else
17.     Troppi geni, impossible risolvere.
18.     VisualizzaGrafo(K)
```

Lis. 3.2: Pseudo codice della funzione ResolveLoop.

```
BackwardAnalysis(x, y)
1.  ▷ Esite la regolazione y ⇒ x
2.  eff(x) = eff(x) − y
3.  ▷ Cerco possibili regolazioni x ⇒ k
4.  if length(eff(x)) = 1 then
5.      ▷ Single Effect Rule
6.      C[eff(x), x] ← 1
7.      ▷ Last Parent Rule
8.  else if ∀ k ∈ eff(x) :  length(eff(k)) = 0 and per(k) ⊂ per(x) then
9.      C[k, x] ← 1
10.     ▷ Ugly Duckling Rule
11. else if ∃ k ∈ eff(x) | k non osservabile and (per(k) ∩ eff(x)) = ∅ then
12.     C[k, x] ← 1
13. else
14.     ▷ Simple Inclusion Rule
15.     if ∃ k | eff(x) = {k, eff(k)} or eff(x) = {−k, −eff(k)} then
16.         C[k, x] ← 1
17.             ▷ Alternative Pathway Rule
18.     else if ∃ k |eff(x) = {−k, eff(k)} or eff(x) = {k, −eff(k)} then
19.         for all t ∈ eff(x) do
20.             C[t, x] ← 1
21.             ▷ Strict Inclusion Rule
22.     else if ∃ k | eff(x) = {k, eff(k), K} or eff(k) = {−k, −eff(k), K}
    then
23.         if ∀ t ∈ K :  (t osservabile and x ∉ eff(t)) or
24.                       (∃ z | t ∈ eff(z) and x ∉ eff(z)) then
25.             C[t, x] ← 1
26. ▷ Inverse Rule
27. for all k ∈ eff(x) | per(k) = x and ∀ t ∈ eff(x) : t osservabile do
28.     C[k, x] ← 1
29. ▷ Cerco possibili regolazioni y ⇒ k
30. K = eff(y) ∩ eff(x)
31. if ∀ k ∈ K :  k osservabile and length(eff(k)) = 0 and per(k) ⊂ per(y)
    then
32.     ▷ Last Parent Rule
33.     C[k, y] ← 1
34. else if ∃ k ∈ K | k non osservabile and (per(k) ∩ eff(y)) = ∅ then
35.     ▷ Ugly Duckling Rule
36.     C[k, y] ← 1
```

Lis. 3.3: Pseudo codice della funzione BackwardAnalysis.

possibile, inoltre, fornire dei parametri opzionali per abilitare/disabilitare alcune funzioni o per settare delle variabili, come ad esempio la soglia θ o il vettore dei pesi assegnati a ciascuna regola d'inferenza ω. In *output*, infatti, l'algoritmo restituisce la matrice di connettività $\mathbf{C}^{n \times n}$, i cui elementi $C[y, x] \neq 0$ rappresentano la regolazione $x \Rightarrow y$ inferita e il cui valore identifica la regola che ne ha permesso l'asserzione. Come accennato in precedenza, un valore $C[y, x] > 0$ identifica la regolazione positiva $x \Rightarrow y$, mentre un valore $C[y, x] < 0$ identifica la regolazione negativa $x \Rightarrow -y$.
Per una panoramica sulle modalità d'invocazione e sui parametri opzionali si rimanda all'Appendice A.
Analizzando la complessità computazionale del core dell'algoritmo (listato 3.1), si può notare che la fase di pre-elaborazione (righe 1–7), cioè inizializzazione dei dati, identificazione delle liste degli effetti osservabili e dei geni perturbatori, richiede $O(n)$ operazioni.
La ricerca delle regolazioni è divisa in due fasi, entrambe le quali si svolgono all'interno di un doppio ciclo che analizza ogni gene, richiedendo quindi $O(n)$ operazioni. Nella prima fase vengono ricercate le regolazioni appartenenti alle categorie:

- *sigle effect rule*: (righe 13–14) che richiede una sola operazione;
- *last parent rule*: (righe 20–22) che richiede $O(n)$ operazioni;
- *ugly duckling rule*: (righe 24–26) che richiede una sola operazione;
- *simple inclusion rule*: (righe 28–29) che richiede $O(n)$ operazioni;
- *strict inclusion rule*: (righe 31–35) che richiede $O(n^3)$ operazioni nel caso peggiore;

Tra le due fasi, ma associabile alla prima, si trova la ricerca di possibili regolazioni attraverso l'*inverse rule*, che richiede $O(n)$ operazioni (righe 37–38).
Nella seconda fase, invece, vengono ricercate le regolazioni appartenenti alle categorie:

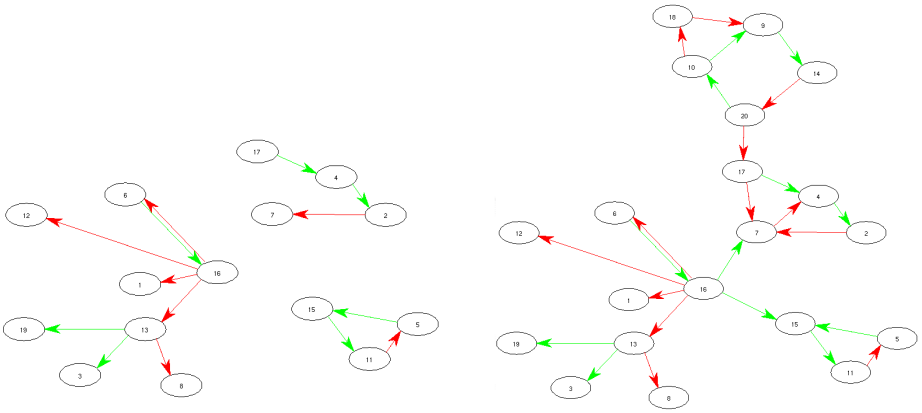
- *alternative pathway rule*: (righe 43–46) che richiede $O(n)$ operazioni;
- *last parent rule*: (righe 50–52) che richiede $O(n)$ operazioni;
- *alternative pathway rule*: (righe 54–55) che richiede $O(n)$ operazioni;

L'identificazione e il conseguente tentativo di risoluzione autonoma di possibili loop all'interno della rete avviene nella prima fase (righe 17–18) e richiede una sola operazione per quanto riguarda l'identificazione, mentre richiede $O(n^2)$ operazioni per la risoluzione (listato 3.2). Ciò è dovuto principalmente al metodo di *backward analysis* illustrato nel listato 3.3.

Si può quindi affermare che nel caso peggiore l'algoritmo richiede un numero di operazioni proporzionale a $O(n^4)$, dovuto all'individuazione delle regolazioni mediante la *strict inclusion rule* della prima fase.

3.6 Fattori limitanti dell'analisi

Nelle sezioni precedenti sono state presentate e discusse le estensioni applicate al framework di partenza. Durante questa fase si è sempre fatto riferimento ad una rete di esempio composta da 20 geni. In figura 3.6 è mostrata la Rete di Regolazione Genica inferita dall'algoritmo e la Rete di Regolazione Reale per la medesima rete.



(a) Rete Inferita. (b) Rete Reale

Fig. 3.6: Rappresentazione grafica della Rete di Regolazione Genica Inferita e della Rete di Regolazione Genica Reale.

A prima vista, sembra che l'algoritmo non sia in grado di asserire tutte le regolazioni che compongono la rete reale, bensì soltanto un sottoinsieme. In realtà, osservando più attentamente la rappresentazione grafica della rete reale (Fig. 3.6(b)), si può notare come i geni 9, 10, 14, 18 e 20 formino un anello di autoregolazione composto da più di 3 geni, che, come descritto in Sezione 3.4, viene correttamente identificato dall'algoritmo ma non disambiguato in maniera autonoma.
Proseguendo nella ricerca delle cause che hanno impedito all'algoritmo di inferire tutte le regolazioni, troviamo i geni non osservabili, cioè quei geni il cui esperimento di knock-out non porta nessuna informazione qualitativa utile all'analisi. È il caso del gene 7 nella rete d'esempio, diventa quindi impossibile asserire regolazioni in uscita dal gene, come ad esempio la regolazione $7 \Rightarrow -4$.
L'uguaglianza dei segni degli effetti osservati nell'esperimento di knock-out del gene 17 e del 4, in particolare del segno positivo sul gene 7, non permette all'algoritmo di identificare la regolazione diretta $17 \Rightarrow 7$, identificandola

invece, come indiretta: $17 \Rightarrow 4 \Rightarrow 2 \Rightarrow 7$.
Infine possiamo notare come l'anello di autoregolazione tra i geni 6 e 16, introduca un elevato fattore d'incertezza sulle regolazioni uscenti dai geni in loop, che non permette di asserire con certezza quale tra i due geni regoli 7 e 15.
È quindi chiaro come i principali fattori che limitano l'asserzione della Rete di Regolazione Genica nella sua interezza siano:

1. geni non osservabili: oltre a perdere le regolazioni dirette uscenti da quei geni, come nell'esempio sopraccitato, in alcuni casi la presenza di almeno uno di essi nella lista degli effetti osservabili di un gene, impedisce di inferire con certezza alcune regolazioni. Se ad esempio la lista degli effetti osservabili del gene x è $\{y, k\}$ con k gene non osservabile, non è possibile sapere se e come k perturbi lo stato di y, impedendo così l'inferenza della regolazione $x \Rightarrow y$.

2. presenza contemporanea di uno o più geni nelle liste degli effetti osservabili dei geni appartenenti ad un anello di autoregolazione risolto in modalità autonoma dall'algoritmo. In questa situazione è difficile capire quale dei geni in loop regoli direttamente gli altri geni. Se ad esempio i geni x, y, k sono in autoregolazione tra di loro e il gene t compare in tutte le loro liste degli effetti osservabili, vista la natura ciclica del loop, non è possibili stabilire con certezza chi regoli direttamente t.

3. loop composti da più di 3 geni o con regolazioni con segno concorde, non risolti dall'algoritmo in maniera automatica.

Capitolo 4

Risultati Sperimentali

Nella Sezione 2.3 sono stati descritti i due dataset utilizzati per testare e validare i risultati ottenuti con l'algoritmo sviluppato in questa tesi. In questo capitolo discuteremo innanzitutto dei valori di default assegnati ai due parametri che caratterizzano maggiormente l'analisi, cioè della soglia θ e del vettore dei pesi associati ad ogni regola d'inferenza ω. In seguito verranno mostrati i risultati ottenuti con entrambi i dataset e confrontati con i risultati ottenuti col framework di partenza, in termini di precisione e richiamo, come anticipato nella Sezione 2.4.

4.1 Parametri d'esecuzione ottimi

Nel corso dei precedenti capitoli si è fatto accenno ai due parametri che maggiormente influiscono sui risultati ottenuti dall'algoritmo: la soglia θ e il vettore dei pesi delle regole d'inferenza ω.
Per quanto riguarda la soglia θ, si è scelto di mantenere i valori di default utilizzati nel framework di partenza, in quanto sono stati utilizzati i medesimi dataset e per poter eseguire dei confronti tra i risultati ottenuti dai due algoritmi, non vi era motivo di apportare modifiche a questi valori.
L'asserzione di quattro ulteriori regole d'inferenza, ha reso necessaria l'introduzione di un vettore dei pesi, ω, che stabilisca una gerarchia tra le regole stesse ed eviti l'inferenza multipla di una singola regolazione, nel caso il peso della regola che ha permesso l'inferenza della regolazione in seconda battuta sia inferiore al peso della regola che ne ha permesso l'inferenza in prima battuta.
La determinazione del vettore ω ottimo ha richiesto il calcolo di precisione e richiamo per ogni singola regola d'inferenza e la determinazione della successione ottima, tale da garantire una precisione non crescente e un richiamo non decrescente all'introduzione di ogni singola regolazione, come mostrato in figura 4.1 e 4.2.
Come si può notare, oltre alle regole d'inferenza mostrate in sezione 3.2, ve

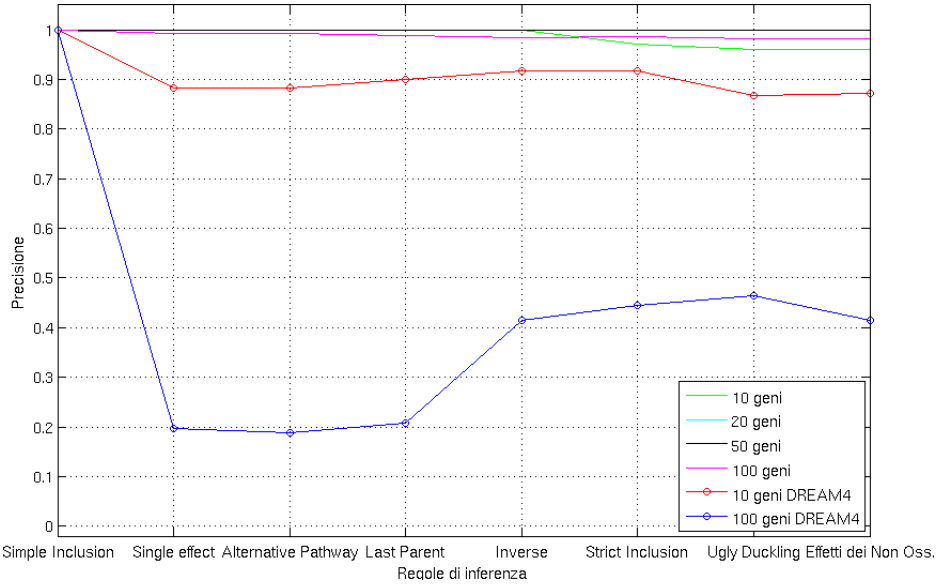


Fig. 4.1: Andamento della Precisione media all'introduzione di ogni regola d'inferenza.

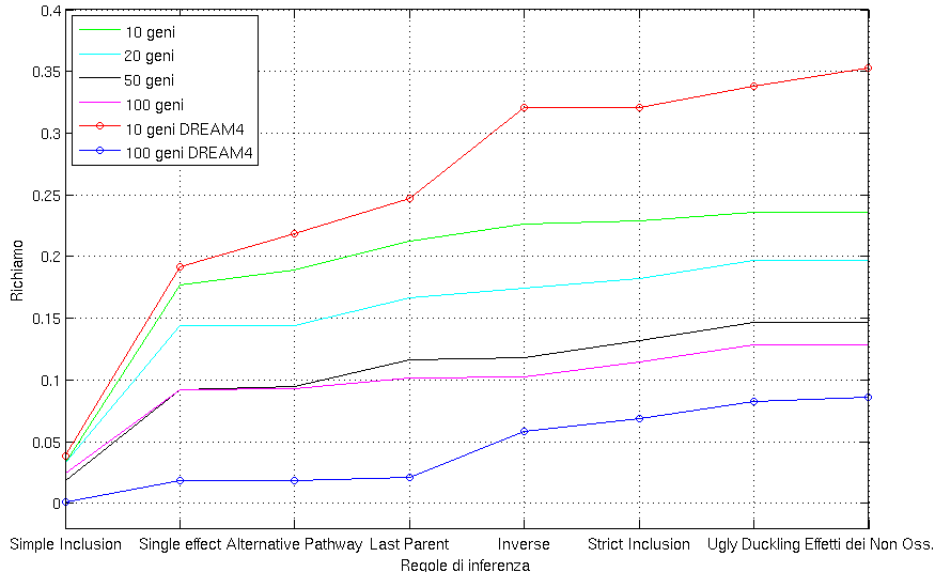


Fig. 4.2: Andamento del Richiamo medio all'introduzione di ogni regola d'inferenza.

n'è un'ulteriore: *Effetti dei geni non osservabili*. Questa non è propriamente una regola d'inferenza, ma identifica le regolazioni inferite per quei geni appartenenti alla categoria *non osservabili*, ma per i quali esistono degli effetti osservabili. A queste regolazioni viene assegnato il peso della regola che ne ha permesso l'asserzione sommato al numero totale delle regole. Ad esempio

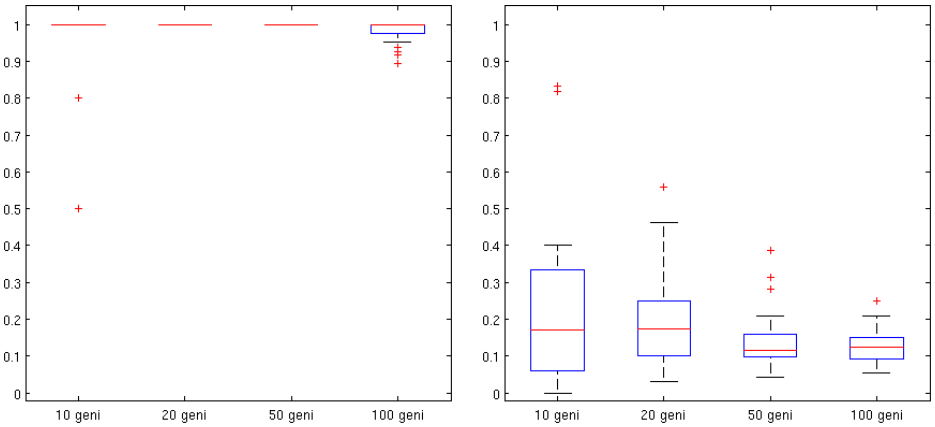
una regolazione appartenente a questa categoria e identificata dalla regola d'inferenza che ha peso 2, nella matrice di connettività che l'algoritmo restituisce in output ha peso 9. Naturalmente, nel caso si tratti una regolazione negativa, il peso della regolazione passa da -2 a -9.
Quest'estensione è attuata soltanto per le reti appartenenti al dataset con rumore, in quanto nel dataset in assenza di rumore non si riscontrano casi di questo tipo, se non in numero molto limitato. Per questo motivo l'estensione della lista dei geni osservabili non àbilitata normalmente, ma richiede l'abilitazione tramite apposito parametro opzionale.
Considerando quindi opzionale l'estensione sopracita, il vettore ω ottimo ottenuto nelle prove eseguite, prevede un'assegnazione dei pesi assoluti così distribuita:

1. *simple inclusion rule*
2. *single effect rule*
3. *alternative pathway rule*
4. *last parent rule*
5. *inverse rule*
6. *strict inclusion rule*
7. *ugly duckling rule*

L'andamento di precisione e richiamo all'introduzione di ogni regola di inferenza, mostrato in figura 4.1 e 4.2, mostra come il vettore ω ottimo garantisca un richiamo non decrescente in tutte le reti considerate e una precisione non crescente in tutte le reti considerate, ad eccezione delle reti del dataset affetto da rumore, soprattutto per quelle composte da 100 geni.
Come si è visto nella sezione 3.5, attraverso l'utilizzo dei parametri opzionali, è comunque sempre possibile settare soglia e vettore dei pesi a valori differenti.

4.2 Dataset in assenza di rumore

L'algoritmo è stato inizialmente testato sul dataset descritto in sezione 2.3.1, composto da esperimenti condotti su 20 differenti reti per ogni taglia: 10, 20, 50 e 100 geni.
In figura 4.3 sono mostrati i boxplot di precisone e richiamo calcolati per il dataset in esame e dai quali si può facilmente notare come la precisone sia pari ad 1 nella maggioranza degli esperimenti, ad indicare come il numero di falsi positivi sia estremamente basso. In particolare, le regolazioni identificate negli esperimenti condotti su reti composte da 20 e 50 geni sono

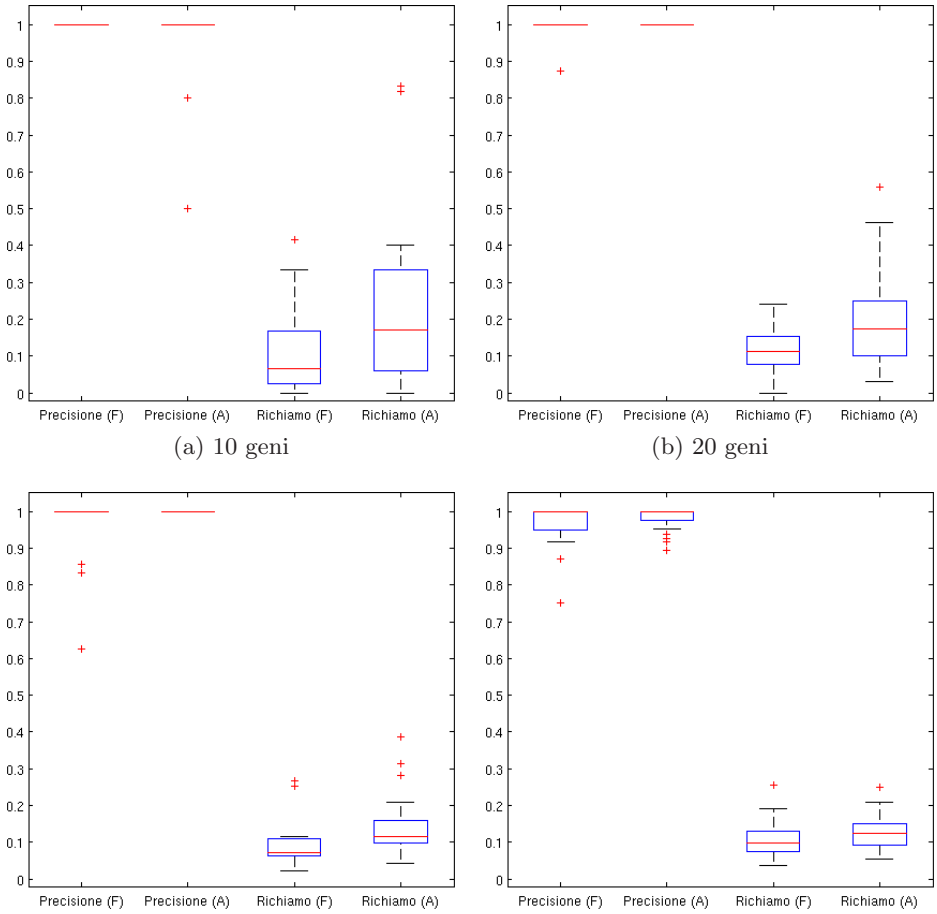


(a) Precisione.

(b) Richiamo.

Fig. 4.3: Boxplot di Precisone e Richiamo per il dataset in assenza di rumore.

sempre esatte. Il richiamo, dal lato opposto, è basso, attestandosi su un valore medio del 18%, con alcuni picchi superiori all'80% nelle reti composte da 10 geni.
Dai boxplot in figura 4.4 si può notare come l'algoritmo sviluppato nel corso di questa tesi, mantenendo un alto livello di precisione, riesca inoltre ad innalzare il livello di richiamo medio. Come accennato in precedenza, questo miglioramento può essere associato alla risoluzione autonoma degli anelli di autoregolazione resa possibile grazie alla considerazione del segno delle regolazioni attuato dall'algoritmo.
Dal lato della complessità computazionale non ci sono sostanziali differenze tra il framework di partenza e l'algoritmo sviluppato in questa tesi. Entrambi, infatti, presentano un numero di operazioni, calcolato nel caso peggiore, proporzionale a $O(n^4)$.



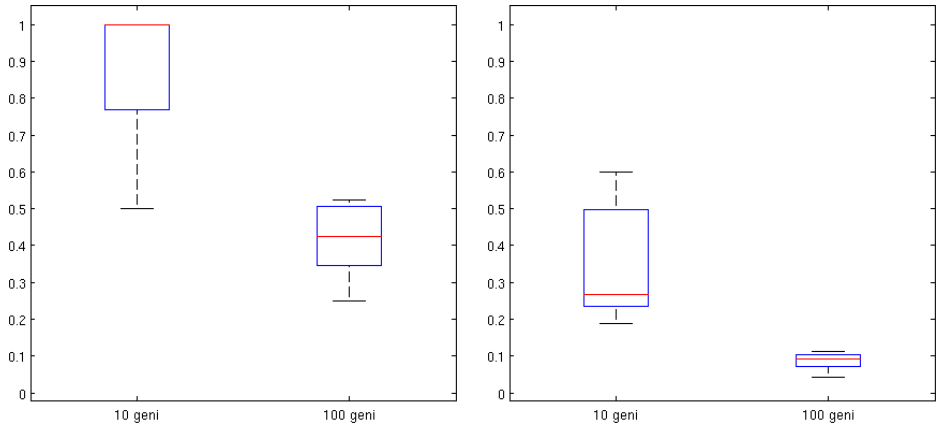
(a) 10 geni (b) 20 geni

(c) 50 geni (d) 100 geni

Fig. 4.4: Boxplot di Precisione e Richiamo calcolati con il framework di partenza (F) e con l'algoritmo sviluppato in questa tesi (A) per le tutte le taglie delle reti appartenenti al dataset in assenza di rumore.

4.3 Dataset con rumore

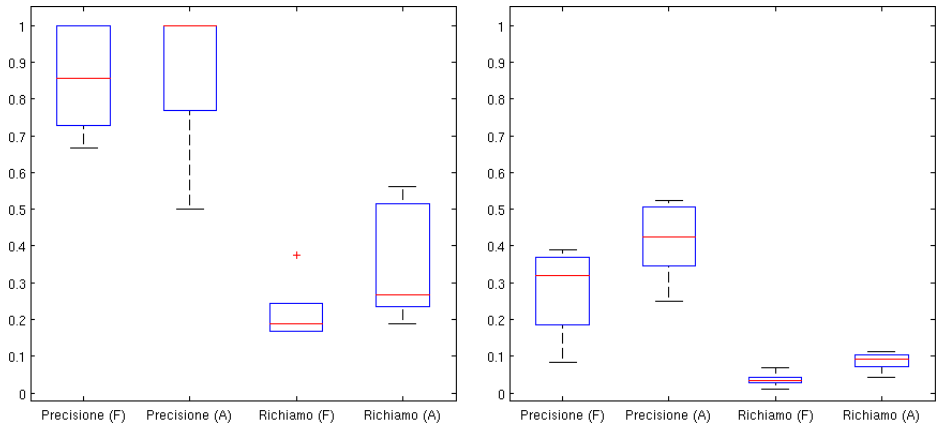
In seguito ai test mostrati nella sezione precedente, l'algoritmo è stato testato su un insieme di esperimenti di gene knock-out estratti dal DREAM4 In Silico Network Challenge del 2009. Il dataset contiene le osservazioni ottenute dalla soppressione sistematica di ogni gene, accompagnati dal profilo di espressione osservato nell'esperimento di wild-type, per 5 reti di 10 e 100 geni. In figura 4.5 vengono mostrati i boxplot di precisione e richiamo ottenuti dall'algoritmo.
Come si può notare, la mediana della precisione per le reti composte da 10 geni assume un valore pari ad 1 anche con questa tipologia di dati, mentre si attesta su un 0,43 per le reti di 100 geni. Per quanto riguarda il richiamo, invece, si può notare come nel caso delle reti composte da 10 geni raggiunga



(a) Precisione. (b) Richiamo.

Fig. 4.5: Boxplot di Precisone e Richiamo per il dataset con rumore (DREAM4).

un valore pari al 27%, ben più elevato del massimo ottenuto con il dataset in assenza di rumore.
Il confronto di precisione e richiamo ottenuti dal framework di partenza e dall'algoritmo sviluppato nel corso di questa tesi, visibile in figura 4.6, mostra come il nuovo algoritmo riesca ad ottenere precisione e richiamo migliori rispetto a quelli ottenuti dal framework di partenza, anche per le reti con 100 geni, nelle quali la precisione non raggiunge livelli ottimi.



(a) 10 geni (b) 100 geni

Fig. 4.6: Boxplot di Precisone e Richiamo calcolati con il framework di partenza (F) e con l'algoritmo sviluppato in questa tesi (A) per le tutte le taglie delle reti appartenenti al dataset con rumore (DREAM4).

Il nuovo algoritmo di Ragionamento Qualitativo sviluppato in questa tesi, grazie all'alto livello di Precisione raggiunto e al tempo d'esecuzione polinomiale, può quindi candidarsi come un buon strumento di pre-elaborazione

per altri algoritmi di Reverse Engineering, in grado di fornire informazioni accurate e attendibili su un sottoinsieme delle regolazioni totali o identificando porzioni della rete di regolazione per le quali è necessario un ulteriore analisi, come ad esempio un anello di autoregolazione, per il quale l'algoritmo non è stato in grado di fornire una risoluzione in maniera autonoma.

Capitolo 5

Conclusioni

In questa tesi, il problema del Reverse Engineering di Reti di Regolazione Genica partendo da esperimenti di DNA microarray è stato studiato da un punto di vista algoritmico.
Dopo aver introdotto il concetto di problema biologico, è stato descritto un algoritmo basato sul paradigma del Ragionamento Qualitativo, sviluppato da F. Sambo nel corso del suo dottorato di ricerca presso l'Università di Padova, in grado di inferire regolazioni dirette tra coppie di geni dall'informazione qualitativa ottenuta dalla sottrazione del profilo di espressione genica dell'esperimento wild-type da ogni esperimento di knock-out eseguito sistemicamente su ogni gene appartenente alla rete.
L'algoritmo descritto, non considerando il segno delle regolazioni tra i geni, fornisce un'informazione incompleta sulle regolazioni inferite e non è in grado di disambiguare eventuali casi di autoregolazione tra i geni (loop).
L'algoritmo sviluppato nel corso di questa tesi estende il framework di partenza introducendo la gestione del segno delle regolazioni, grazie alla quale è possible inferire due tipologie di regolazione: positiva (attivazione) e negativa(inibizione). Questa estensione ha permesso, inoltre, la definizione di un metodo in grado di risolvere autonomamente i casi di autoregolazione individuati.
Un'ulteriore estensione applicata al framework di partenza riguarda il numero di regole d'inferenza: sono state infatti formulate quattro ulteriori regole per permettere l'asserzione di un numero maggiore di regolazioni. Infine è stata introdotta la possibilità di visualizzare una rete di regolazione genica come un grafo diretto in cui i nodi rappresentano i geni, mentre gli archi orientati rappresentano l'azione regolatoria tra nodi diversi. Sfruttando questa caratteristica delle reti di regolazione, l'algoritmo fornisce una rappresentazione grafica dei geni appartenti ad un loop impossibile da risolvere dall'algoritmo in modalità autonoma.
Grazie alla modifica delle strutture dati contenenti le informazioni qualitative ottenute dai dati analizzati, è stato possibile mantenere il tempo d'ese-

cuzione polinomiale, più precisamente mantenendolo proporzionale a $O(n^4)$. L'alto livello di Precisone raggiunto dal nuovo algoritmo di Ragionamento Qualitativo, unito al tempo d'esecuzione polinomiale, rendono l'algoritmo un buono strumento di preprocessing, in grado di fornire informazioni accurate e attendibili su un sottoinsieme delle regolazioni totali, permettendo ad un secondo algoritmo di ridurre lo spazio di ricerca, ad esempio ai soli nodi appartenenti ad un loop, per il quale l'algoritmo non è stato in grado di fornire una risoluzione in maniera autonoma.
L'algoritmo, infatti, incontra una certa difficoltà nel ricostruire una Rete di Regolazione Genica nella sua interezza, principalmente a causa della mancanza di informazioni sugli effetti osservabili per alcuni geni, cioè quelli appartenenti alla categoria dei *non osservabili*. La limitazione del metodo di risoluzione autonoma dei loop ai casi con massimo 3 geni in autoregolazione, assieme all'impossibilità di asserire con certezza le regolazioni in uscita dai geni in loop, costituisco i due ulteriori fattori che impediscono la ricostruzione completa delle reti di regolazione analizzate.
Alcuni sviluppi futuri possono essere quindi ricondotti al miglioramento e potenziamento del metodo di risoluzione autonoma, aumentando il numero massimo di geni appartenenti ad un anello di autoregolazione e identificando correttamente le regolazioni uscenti dai geni in loop, verso gli altri geni appartenenti alla rete.

Bibliografia

[1] AT&T. Open source graph visualization software. URL: http://www.graphviz.org/.

[2] P.A. Levene and L.W. Bass. Nucleic acids. *Chemical Catalogue Co*, 1931.

[3] J.D. Watson and F. Crick. A Structure for Deoxyribose Nucleic Acid. *Nature*, 171:737–738, Aprile 1953.

[4] J.D. Watson and F. Crick. Genetical implications of the structure of deoxyribonucleic acid. *Nature*, 171:964–967, Maggio 1953.

[5] Francis Crick. Central Dogma of Molecular Biology. *Nature*, 227:561–563, Agosto 1970.

[6] The MathWorks, Inc. Matlab - the language of technical computing. URL: http://www.mathworks.com/, 1984.

[7] S.A. Kauffman. The origins of order: self-organization and selection in evolution. *Oxford University Press*, 1993.

[8] M. Schena, D. Shalon, RW. Davis, and PO. Brown. Quantitative monitoring of gene-expression patterns with a complementary-dna microarray. *Science*, 270(1):467–470, 1995.

[9] M.I. Arnone and E.H. Davidson. The hardwiring of development: organization and function of genomic regulatory systems. *Development*, 124:1851–1864, 1997.

[10] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, Giugno 1998.

[11] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, Ottobre 1999.

[12] John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel. Performance measures for information extraction. In *Proceedings of DARPA Broadcast News Workshop*, pages 249–252, 1999.

[13] G. Ruvkun. Molecular biology. glimpses of a tiny rna world. *Science*, 294(5543):797–799, 2001.

[14] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabasi. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, August 2002.

[15] B. Zupan, J. Demsar, I. Bratko, P. Juvan, J. A Halter, A. Kuspa, and G. Shaulsky. Genepath: a system for automated construction of genetic networks from mutant data. *Bioinformatics*, 19(3), 2003.

[16] B. Zupan, J. Demsar, I. Bratko, P. Juvan, J. A Halter, A. Kuspa, and G. Shaulsky. Genepath: a system for inference of genetic networks and proposal of genetic experiments. *Artificial Intelligence in Medicine*, 29(1-2), 2003.

[17] E.P. Diamandis. Mass spectrometry as a diagnostic and a cancer biomarker discovery tool. *Molecular & Cel Proteomics*, 2004.

[18] Lawrence Hunter. Life and Its Molecules. *AI Magazine*, 25(1):9–22, Spring 2004.

[19] Leon Peshkin. Matlab - graphviz interface. URL: http://www.mathworks.com/matlabcentral/fileexchange/4518-matlab-graphviz-interface, Febbraio 2004.

[20] Michael Molla, Michael Waddell, David Page, and Jude Shavlik. Using Machine Learning to Design and Interpret Gene-Expression Microarrays. *AI Magazine*, 25(1):23–44, Spring 2004.

[21] R. Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118:4947–4957, 2005.

[22] Alon and Uri. Network motifs: theory and experimental approaches. *Nat. Rev. Genet.*, 8(6):450–461, June 2007.

[23] Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego Di Bernardo. How to infer gene networks from expression profiles. *Mol. Syst. Biol.*, 3(78), February 2007.

[24] B. Di Camillo, G. Toffolo, and C. Cobelli. A gene network simulator integrating boolean regulation in a continuous dynamic model. In *Proc. of the 5th European Conference on Computational Biology - ECCB '06, Eilat, Israel*, 2007.

[25] Neil Hall. Advanced sequencing technologies and their wider impact in microbiology. *J Exp Biol*, 210(9):1518–1525, May 2007.

[26] Nicola Soranzo, Ginestra Bianconi, and Claudio Altafini. Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks: synthetic versus real data. *Bioinformatics*, 23(13):1640–1647, July 2007.

[27] A. Corradin, B. Di Camillo, G. Toffolo, and C. Cobelli. In silico assessment of four reverse engineering algorithms: role of network complexity and multi-experiment design in network reconstruction and hub detection. In *ENFIN-DREAM Conference Assessment of Computational Methods in Systems Biology, Madrid, April 28-29*, 2008.

[28] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229–239, 2009.

[29] Gustavo Stolovitzky, Robert J. Prill, and Andrea Califano. Lessons from the DREAM2 challenges: A community effort to assess biological network inference. *Annals of the New York Academy of Sciences*, 1158(1):159–195, 2009.

[30] Francesco Sambo. Advanced algorithms for genomic data analysis. Master's thesis, Università degli Studi di Padova, Febbraio 2010.

[31] Bellman and Richard E. *Adaptive control processes - A guided tour*. Princeton University Press, Princeton, New Jersey, U.S.A., 1961.

Appendice A

Manuale d'uso

A.1 analyzeKO.m

AnalyzeKO è lo strumento che, partendo da profili d'espressione genica di esperimenti di gene knock-out e da un profilo di riferimento (wild-type), ricostruisce la Rete di Regolazione Genica.

Sintassi

```
analyzeKO(data,wt)
analyzeKO(...,'NomeParametro',ValoreParametro,...)
[matrix, unobserved, leaf] = analyzeKO(...)
```

Descrizione

`analyzeKO(data,wt)`
Esegue un'analisi della matrice *data* contenente i livelli di espressione genica ottenuti dagli esperimenti di gene knock-out, sottraendo ad essa il vettore colonna *wt*, livello di espressione genica nell'esperimento di wild-type.
Viene fornita in output una matrice dove i valori diversi da 0 idenficano le regolazioni inferite durante l'analisi. I differenti valori presenti nella matrice permettono di individuare quale regola ha permesso l'inferenza della regolazione.
`analyzeKO(...,'NomeParametro',ValoreParametro,...)`
Permette di impostare una proprietà al valore specificato. Una descrizione delle proprietà utilizzabili ed il loro valore di default è mostrato nella sezione seguente.
`[matrix, unobserved, leaf] = analyzeKO(...)`
Ritorna in output oltre alla matrice contenente le regolazioni individuate, anche la lista dei geni classificati come *non osservabili* durante l'analisi. Viene restituita anche una lista dei geni la cui lista di effetti osservati è vuota.

Parametri Opzionali

- *Verbose*: abilita la scrittura a video dell'analisi, indicando la lista degli effetti osservati per ogni esperimento di gene knock-out, le regolazioni inferite e i loop individuati. Questa opzione è disabilitata di default.

- *Threshold*: indica la soglia minima di variazione tra il livello di espressione del gene, misurato nell'esperimento di gene knock-out e di wild-type. Se non specificata è pari a 0.05.

- *ExtendObservable*: estende la lista dei geni osservabili anche ai geni che dovrebbero essere inseriti nella categoria non osservabili, ma la cui lista di effetti osservabili è diversa dall'insieme vuoto. Questa opzione è disabilitata di default.

- *FileOutput*: abilita la scrittura dei risultati dell'analisi in un file di testo, nella directory *output*. Questa opzione è disabilitata di default.

- *DrawGraphs*: abilita la rappresentazione grafica della matrice dell'informazione qualitativa ottenuta dalla sottrazione del wild-type da ogni esperimento di knock-out. Abilita, inoltre, la visualizzazione del grafo della Rete di regolazione inferita durante l'analisi e, nel caso in cui l'algoritmo non riesca a disambiguare eventuali casi particolari, abilita la rappresentazione della porzione di grafo in autoregolazione.

- *ReverseList*: abilita la scrittura in output o su file della lista dei geni regolatori per ogni gene. Se *Verbose* e *FileOutput* sono disabilitate, non verrà eseguito nessuna operazione.

- *RegWeight*: permette di impostare i pesi delle regolazioni per la matrice di output, indicandoli secondo l'ordine:

  - single effect rule: valore di default: 2
  - simple inclusion rule: valore di default: 1
  - strict inclusion rule: valore di default: 6
  - last parent rule: valore di default: 4
  - inverse rule: valore di default: 5
  - alternative pathway rule: valore di default: 3
  - ugly duckling rule: valore di default: 7

A.2 viewMat.m

ViewMat fornisce una rappresentazione grafica per le matrici di espressione genica ottenute da esperimenti di DNA-Microarray.

Sintassi

```
viewMat(data)
viewMat(data,th)
```

Descrizione

ViewMat fornisce in output un'immagine della matrice *data*, dove ogni riga rappresenta un gene, mentre ogni colonna un esperimento.
Nel caso in cui *data* contenga livelli di espressione genica ottenuti da esperimenti di DNA-microarray, l'immagine è in scala di grigi e l'intensità è proporzionale al livello di espressione del gene, dall'assenza (bianco) al massimo livello misurato (nero).
Nel caso in cui *data* contenga l'informazione qualitativa ottenuta dalla sottrazione del wild-type da ogni esperimento di knock-out, l'intensità è proporzionale al livello di espressione del gene in relazione al wild-type, dal massimo livello di sotto-espressione misurato (rosso), al massimo livello di sovre-espressione misurato (verde), passando per un'eguale livello (bianco).

Parametri Opzionali

- *th*: è il valore di soglia utilizzato per eliminare i valori compresi tra $-th$ e th, che non portano nessuna informazione utile. Il valore di default è 0.05.