*<<L'uomo è confinato nei limiti angusti del corpo,*

*come in una prigione, ma la matematica lo libera,*

*e lo rende più grande dell'intero universo. [...]*

*Sballottato qua e là, senza meta, dalla tempesta*

*delle passioni, la matematica gli restituisce*

*la pace interiore , risolvendo armoniosamente*

*i moti opposti dell'anima, e riconducendola,*

*sotto la guida della ragione, all'accordo e all'armonia.>>*

*Pietro Ramo, Institutiones dialecticae*

# Introduzione

La tecnologia basata su Microarray costituisce uno degli approcci recenti più promettenti nell'ambito della genetica high-throughput e fornisce l'opportunità di studiare i livelli di espressione dei geni su scala genomica. Attualmente è possibile monitorare con un solo Microarray migliaia di geni, finanche decine di migliaia, tramite la misurazione dell'intensità della fluorescenza dell'mRNA ibridato. Ciò rende possibile la misurazione dei livelli di espressione dell'intero insieme dei geni di un organismo.

I Microarray applicati alla genetica funzionale e all'analisi delle mutazioni geniche permettono di determinare migliaia di livelli di espressione in centinaia di condizioni, consentendo la potenziale comprensione dei processi genici alla base delle patologie genetiche.

Gli esperimenti statici di espressione analizzano campioni relativi a molti individui. Tali campioni spesso possono essere considerati istantanee della progressione di alcune patologie genetiche degenerative, come il cancro. È possibile determinare un ordine temporale per tali campioni? Ovvero: è possibile ricavare da dati statici di espressione genica le soggiacenti dinamiche temporali? Tale eventualità consentirebbe una miglior comprensione delle dinamiche della patologia e l'identificazione dei geni associati alla sua evoluzione, e senz'altro darebbe adito a molte altre possibili applicazioni.

È possibile trovare una prima formulazione del problema nel lavoro di Magwene et Al.[9], ma è nel lavoro di Anupam Gupta e Z. Bar-Joseph[2] dove per la prima volta si prova formalmente che utilizzando un modello per la dinamica del livello di espressione di un singolo gene, che tiene conto di alcune assunzioni verosimili sulla sua dinamica, è possibile ricostruire l'ordinamento corretto di un data set di espressione statica, risolvendo un'istanza del problema del commesso viaggiatore (TSP)

Nello studio qui esposto si è usato il lavoro di Anupam Gupta e Z. Bar-Joseph[2] come punto di partenza. Si è effettuata una nuova validazione sperimentale del metodo proposto. Come notato anche nel lavoro di riferimento si verifica la tendenza del metodo a riordinare in blocchi, permutati tra loro e in verso di percorrenza sia diretto che inverso rispetto all'ordinamento "vero", ossia quello corrispondente all'effettiva dinamica temporale. Si sono quindi realizzate due metodologie algoritmiche per la risoluzione di questo aspetto: la prima tramite bootstrap, la seconda tramite riassetto dei

blocchi con l'analisi della derivata. Mentre la prima si è rivelata, in seguito a valutazioni su dati simulati, incapace di migliorare le prestazioni del metodo originario, la seconda ha evidenziato la capacità, per livelli contenuti di rumore sui dati, di fornire ordinamenti più vicini all'ordinamento "vero". Si è infine applicato il metodo di Gupta Bar-Joseph arricchito con il riassetto dei blocchi a dati reali: un data set ricavato con lo shuffle degli array corrispondenti alle colonne di una matrice di espressione dinamica ( quindi si è ricreato un data set statico con ordine originario noto), e un secondo data set proveniente da esperimenti effettuati su 112 pazienti affetti da LLC (leucemia linfatica cronica). Nel caso del primo data set il metodo del riassetto dei blocchi tramite analisi della derivata ha mantenuto l'ordinamento ottenuto con il metodo deterministico fornito da [2], rivelatosi in effetti l'ordine corretto; per il secondo data set invece si è effettuato un confronto con i parametri prognostici utilizzati in ambito clinico come marker. In questo caso si è evidenziata una miglior concordanza tra l'ordinamento temporale ottenuto e i parametri prognostici nel caso di metodo arricchito con il riordinamento dei blocchi rispetto, al solo metodo di riordinamento dei soggetti.

# Indice

# Capitolo 1

# Stato dell'arte

Estrazione di dinamiche temporali da dati statici di espressione genica nel cancro

Gli esperimenti statici di espressione genica analizzano campioni provenienti da molti individui. Tali campioni possono essere visti come istantanee dell'evoluzione temporale di alcune malattie come ad esempio il cancro. La domanda alla quale si vuole dare risposta nello studio condotto da Gupta e Ziv Bar-Joseph [1] è se sia possibile o meno determinare un ordinamento temporale per questi soggetti. L'ordinamento che ne scaturirebbe potrebbe consentire una miglior comprensione delle dinamiche della malattia e, con l'ausilio di alcune metodologie messe a punto per esperimenti dinamici di espressione genica, permettere di individuare i geni associati alla sua evoluzione e chiarire i meccanismi di regolazione genica che ne sono a capo.

Nel lavoro di Gupta e Ziv Bar-Joseph [1] (dal quale si attingono molte delle informazioni proposte in questo capitolo) viene adottato un modello relativo alla dinamica dei livelli di espressione di un singolo gene, attraverso il quale è possibile ricostruire il corretto ordinamento per i data set di espressione statici risolvendo un'istanza del problema del commesso viaggiatore (TSP). Sempre nel lavoro [1] si è realizzato un algoritmo che combina l'euristica TSP con un modello probabilistico per dedurre l'ordinamento temporale dei dati ottenuti da esperimenti con microarray. Questo algoritmo costruisce curve probabilistiche continue per rappresentare i profili di espressione ed è quindi in grado di considerare il rumore e le differenze di espressione dovute al background individuale permettendo di ottenere una ricostruzione temporale accurata per i dati di espressione genici umani. Applicando tale metodo ai dati di espressione relativi a pazienti affetti da cancro gli autori hanno provato che

l'ordinamento temporale ottenuto con l'algoritmo è in accordo con i tempi di sopravvivenza.

I dati statici di livelli di espressione sono più del 60% di tutti i dati ottenuti tramite esperimenti su microarray. Da tali esperimenti spesso si ottengono istantanee dell'attività di alcuni sistemi o del livello di progressione di alcune malattie. Per esempio, gli esperimenti relativi al livello di espressione genica su pazienti affetti da cancro misurano il livello di espressione dei geni in decine o centinaia di individui. Tra questi individui cambia il tempo di insorgenza della malattia. Dal momento che il cancro progredisce nel tempo le misurazioni di espressione genica relative ad un paziente possono essere interpretate come istantanee di uno specifico stadio dell'evoluzione della malattia.

In molti esperimenti volti a monitorare l' espressione genica il fine è individuare i geni comuni che vengono up-down regolati in una specifica condizione (ad esempio una specifica tipologia di neoplasia maligna) [2]. La mancanza di informazioni temporali relative alla durata della malattia negli individui studiati può complicare questa analisi. Le differenze relative al tempo di insorgenza della patologia possono comportare differenze nei livelli di espressione per un numero considerevole di geni coinvolti nella malattia considerata, rendendo difficile l'identificazione di un consensus set. Dal momento che spesso varia anche il background del livello di espressione tra gli individui, è ancor più difficile individuare tale insieme di consenso e stabilire se le differenze sono dovute alla malattia o ad altri fattori.

Nonostante alcuni studi recenti siano focalizzati alla misurazione della progressione del cancro utilizzando microarray, tali studi sono limitati a modelli animali o linee cellulari [3]. La maggior parte dei dati di espressione di soggetti affetti da cancro è statica e tali dati vengono utilizzati per molti studi di follow-up e per propositi di classificazione dei geni.

Come detto in precedenza, una questione interessante che ci si può porre è se sia o meno possibile ricostruire l'ordinamento temporale per un insieme di campioni-soggetti basandosi sui loro valori di espressione statica. Tale informazione temporale può permette di ricostruire le dinamiche di progressione del cancro e l'identificazione dei geni caratterizzati da comportamento significativo (geni up-down regolati). Un profilo temporale per i geni può inoltre consentire una migliore classificazione. I geni che si

riconoscono variare in una delle classi considerate sulla base di differenze temporali possono essere utilizzati da un classificatore che tenga conto delle dinamiche.

Lo studio [1] propone due importanti contributi:

1) gli autori danno prova formale, per la prima volta, del fatto che adottando un modello ragionevole per la dinamica dei livelli di espressione per un singolo gene, è possibile ricostruire il corretto ordinamento temporale relativo a dati statici di espressione risolvendo un caso particolare del problema del commesso viaggiatore (TSP), per un approfondimento si rimanda all'Appendice al Capitolo corrente "The Travelling Salesman Problem (TSP)". Questo risultato si basa sul fatto che ogni esperimento microarray misura i livelli di espressione di migliaia di geni e che due campioni temporalmente vicini verosimilmente mostreranno valori di espressione simili per la maggior parte di questi geni.

2) Nello studio si realizza un algoritmo che combina un'euristica TSP con un modello probabilistico per ottenere l'ordinamento temporale degli esperimenti su microarray e per utilizzare questo ordinamento per ricostruire i profili temporali a partire dai dati statici di espressione. Utilizzando curve probabilistiche continue per rappresentare i profili di espressione è possibile tener conto del rumore e delle differenze di espressione dovute al background individuale permettendo di ottenere una ricostruzione temporale accurata per i data di espressione umani.

Utilizzando dati di espressione relativi a 50 pazienti affetti da glioma (vedi Appendice B."Il glioma") gli autori hanno dimostrato che l'algoritmo realizzato si rivela in grado di ricostruire un ordinamento temporale che si accorda molto bene con i tempi di sopravvivenza. Inoltre una classificazione basata sull'ordinamento ottenuto offre delle prestazioni nettamente migliori rispetto ad altri classificatori usualmente utilizzati per questo tipo di esperimenti

## 1.1 Ordinamento deterministico

Per iniziare gli autori hanno considerato dati privi di rumore, mentre nel seguito il metodo si estenderà in modo da considerare anche dati rumorosi. Innanzitutto si sono effettuano alcune assunzioni di base riguardanti le dinamiche dei livelli di mRNA

relativi ad ogni singolo gene. Nella trattazione esposta nello studio si assume che un gene che giochi un ruolo significativo in una specifica malattia non cambi spesso la direzione della sua traiettoria di espressione. Cioè che se il livello di espressione di un gene ad un istante temporale stia crescendo allora all'istante temporale successivo molto probabilmente questo continui a crescere o tutt'al più si mantenga stazionario. Sia *p* sia la probabilità che la traiettoria del livello di espressione di un gene non cambi direzione tra due istanti temporali. Nonostante un gene possa cambiare la traiettoria di espressione più volte, si assume che in presenza di un numero sufficiente di istanti temporali, o indifferentemente di soggetti corrispondenti a istanti temporali non noti, allora la probabilità che l'evento si realizzi sia $(1-p) < {}^{1}/_{2}$ per ogni istante temporale. Questa assunzione è supportata da osservazioni effettuate su diversi set di serie temporali di dati relativi al progresso di una malattia. Si è visto che per tali data set una percentuale esigua dei geni cambiava direzione più di una volta [4] [5]. Un'altra assunzione è che ad ogni istante temporale un gene si mantenga allo stesso livello del precedente istante temporale di campionamento con probabilità *q* e che quindi vari (cresca o decresca in dipendenza dalla direzione) con probabilità (1-q). Un valore basso per *q* è indice che la maggior parte dei geni non variano nel tempo.

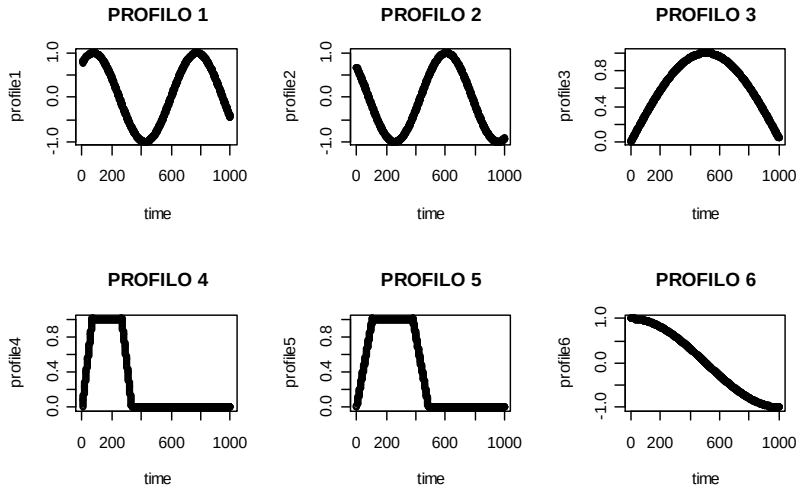


**Figura 1.1.** Alcuni esempio di profili temporali di espressione per geni significativi che tengano conto delle due assunzioni:
1) si assume che la traiettoria del livello di espressione di un gene significativo non cambi spesso direzione e che per pochi geni il cambio di direzione avvenga più di una volta.
2) Si assume che ad ogni istante temporale il livello di espressione di un gene si mantenga costante con probabilità q e invece che cambi direzione con probabilità 1-q.

Alcuni possibili profili di espressione per un gene con comportamento significativo che tengono conto di queste due assunzioni sono rappresentati in Figura 1.1.

Dal momento che la traiettoria del gene in Figura 1.2 (a) prima cresce e poi decresce se si riordinano i valori misurati basandoci sulla loro distanza, si ritrova il profilo rappresentato in Figura 1.2 (b). Come evidente questo ordinamento corrisponde alla somma delle distanze minore, ma non combacia assolutamente con il profilo originale. Fortunatamente in esperimenti con i microarray si dispone di profili relativi a migliaia di geni. Come mostrano le Figure 1.2 (c), 1.2 (d), 1.2 (e) quando si esaminano profili di espressione relativi a più geni, l'ordinamento corretto genera uno score relativo alle distanze migliore se confrontato con un ordinamento che massimizzi uno score per ognuno dei profili preso singolarmente. Gli autori quindi concludono che, dal momento che i geni considerati negli studi effettuati con microarray sono solitamente migliaia, l'ordinamento che minimizza lo score verosimilmente corrisponde all'ordinamento corretto.

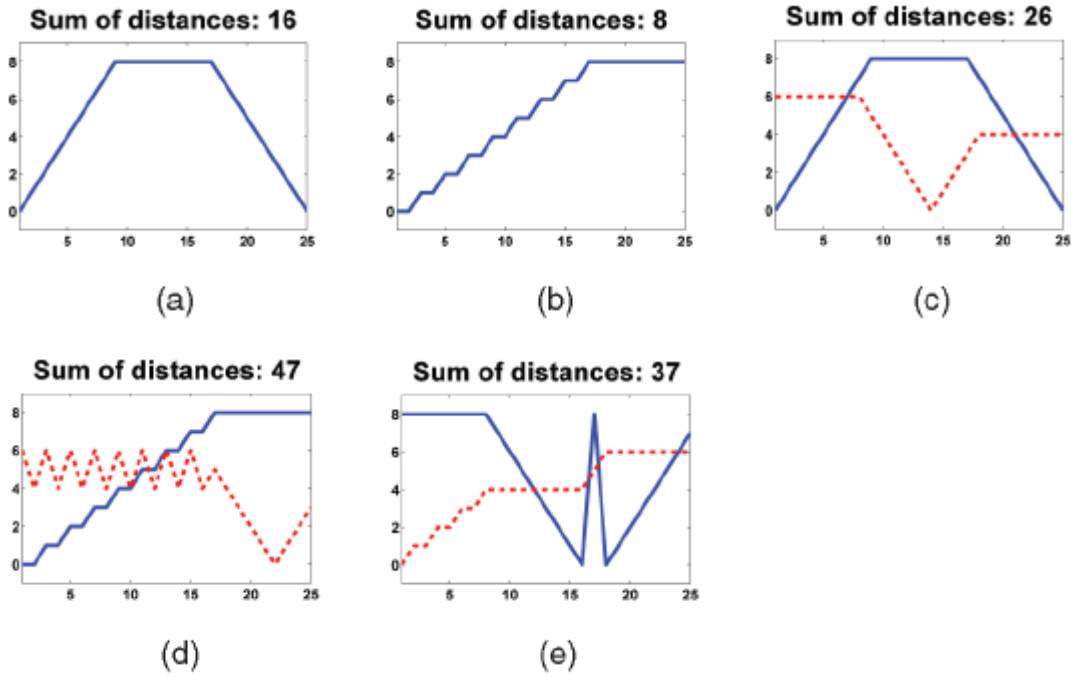



**Figura 1.2.** Riordinamento basato sui valori di uno o due geni. Ogni figura rappresenta una curva campionata uniformemente costituita da 25 campioni. Se si considera un singolo gene (a), il migliore ordinamento (b), cioè quello che minimizza la distanza, non combacia con la curva corretta. Invece quando si considerano due geni (c) l'ordinamento corretto permette di ottenere uno score inferiore se confrontato con gli ordinamenti ottimi per ogni singolo gene (d) e (e). Dal momento che i microarray generano dati relativi a migliaia di geni, verosimilmente un ordinamento con lo score più basso combacerà con l'ordinamento corretto.

### Modello per un singolo gene

Gli autori iniziano presentando un semplice modello per un singolo gene utilizzando una variabile random $Z(t)$, caratterizzata dal comportamento di seguito esposto. Come visto precedentemente, ad ogni istante temporale il gene si mantiene allo stesso livello di espressione con probabilità q, e cambia valore con probabilità (1-q); p è la probabilità che, supponendo che il gene cambi valore, non cambi direzione e quindi (1-p) la probabilità che cambi direzione (ovviamente se non si verifica la condizione di non variazione).

Per modellizzare questo processo vengono introdotte due variabili random:

- $M(t) \in \{0,1\}$, che può essere pensata come indicatrice di "movimento".

- $\Delta(t) \in \{-1,1\}$ , che può essere pensato come indicatrice di "cambio di direzione".

con le seguenti corrispondenti distribuzioni di probabilità:

- M(t) =1 → indica una variazione nel livello di espressione del gene, quindi si ha: $\mathbf{Pr}$ [M(t) = 1] = $1 - q$.
- M(t) =0 → indica che il livello di espressione del gene si mantiene costante: $\mathbf{Pr}$ [M(t) = 0] = $q$.
- $\Delta(t)$ =1 → indica che se il valore del livello di espressione del gene cambia, esso continua a muoversi nella stessa direzione (continua a crescere o decrescere). La probabilità corrispondente all'evento è data da: $\mathbf{Pr}$ [$\Delta(t)$ =1 | M(t)=1] = $p$.
- $\Delta(t) = -1$ → indica un cambio nella direzione, ovviamente ammesso che ci sia una variazione del livello di espressione. La probabilità che avvenga un cambio e che la direzione non vari è data da: $\mathbf{Pr}$ [$\Delta(t) = -1$ | M(t)=1] = $1-p$.

Definendo $D(t) \in \{-1,1\}$ come la "direzione" di movimento del gene al tempo t:

$$D(t+1) = \begin{cases} D(t) * \Delta(t+1) & \text{se M}\,(t+1) = 1 \\ D(t) & \text{se M}\,(t+1) = 0 \end{cases}$$

E quindi si ha che:

$\mathbf{Pr}$ [D $(t+1)$ = D(t) | M (t+1) = 1] = $p$

rappresenta la probabilità che la direzione non vari condizionata al fatto che si verifichi una non costanza del livello di espressione.

Sebbene si siano assunte $M(t)$ e $\Delta(t)$ indipendenti tra loro e indipendenti da tutte le altre $M(t')$ e $\Delta(t')$, non è vero che le direzioni D(t) siano indipendenti tra loro nel tempo, infatti se $p$, ovvero la probabilità che la direzione di variazione si mantenga, è molto vicino a 1, allora D(t) sarà verosimilmente la stessa di D($t-1$).

Ora è nello studio si prosegue definendo la dinamica della variabile random Z(t):

$$Z(t+1) = Z(t) + M(t+1) * D(t+1) = Z(0) + \sum_{\tau=1}^{t+1} M(\tau) * D(\tau).$$

Dove si assume che sia noto il valore iniziale Z(0) per il gene e D(0) per la direzione iniziale del movimento. L'evento Z(t +1) = Z(t), corrispondente all'evento "il valore del livello di espressione del gene all'istante t +1 è lo stesso dell'istante precedente", si verifica con probabilità q.

**Modello per n geni**

In generale vengono considerati vettori che misurano l'espressione di n geni indipendenti:

$\mathbf{Z}(t) = < Z_1(t), Z_2(t),..., Z_n(t) >$,

per ogni istante temporale t e dati campionati a tutti gli istanti temporali $t \in \{1,2, \dots ,T\}$ con $T \in \mathbb{Z}$. È possibile formulare alcuni teoremi, per la dimostrazione dei quali si rinvia il lettore a [1]:

❖ **Teorema 1.1 (One-Step Theorem):**

Supponendo che i valori delle probabilità p e q siano tali per cui $q < 1$ e $\frac{1}{2} < p \leq 1$. Allora la disuguaglianza:

$$\|\boldsymbol{Z}(t+1) - \boldsymbol{Z}(t)\|_1 < \|\boldsymbol{Z}(t+2) - \boldsymbol{Z}(t)\|_1$$

vale simultaneamente per tutti gli istanti temporali $t \in \{1,2, \dots ,T\}$ con $T \in \mathbb{Z}$, con probabilità almeno $1 - Te^{-O(n(1-q))}$.

Dal Teorema 1.1 segue:

❖ **Teorema 1.2 (The General Theorem):**

Supponendo che i valori delle probabilità p e q siano tali per cui $q < 1$ e $\frac{1}{2} < p \leq 1$. Allora la disuguaglianza:

$\|\boldsymbol{Z}(t+1) - \boldsymbol{Z}(t)\|_1 < \|\boldsymbol{Z}(t+k) - \boldsymbol{Z}(t)\|_1$ , con $k \geq 2$ ,

vale simultaneamente per tutti gli istanti temporali t ∈ {1,2, … ,T} con T ∈ ℤ, con probabilità almeno $1-\mathrm{T}e^{-O(n(1-q))}$.

La dimostrazione del Teorema 1.2 è legata all'assunzione che i geni abbiano comportamento indipendente, la qual cosa è un assunzione molto pesante. Comunque, il teorema può essere facilmente esteso ai casi in cui i geni appartengano a m gruppi di dimensione approssimativamente uguale, che siano altamente correlati al loro interno e sufficientemente indipendenti gli uni dagli altri (*cluster*). Dal momento che solitamente il numero di *cluster* distinti negli esperimenti di espressione, quindi di pattern dinamici peculiari, è molto più elevato di log(T), questo risultato può essere applicato a molti data set.

Dal teorema 1.2 segue una conclusione fondamentale:

❖ **Teorema 1.3 (Optimality of TSP Solution):**

Con probabilità molto elevata, l'unico cammino del commesso viaggiatore minimo che passa per tutti i vertici V = {Z(t)| $1 \leq t \leq T$} è il percorso $P^*$=<**Z**(1),Z(2),…,**Z**(T)>, dove i vertici corrispondono agli istanti temporali (o soggetti) e gli archi sono le distanze tra i livelli di espressione ad essi relativi.

Ovvero, finché il numero di campioni indipendenti n è molto più elevato di log(T), dove T è il numero di soggetti-campioni (ed è questo il caso di esperimenti di espressione genica) esiste un'unica soluzione del *TSP path* e questa ricostruisce l'ordinamento corretto, con probabilità molto alta.

Di questa fondamentale conclusione gli autori forniscono la semplice dimostrazione riportata nel seguito.

Per prima si nota che la lunghezza del *path* $P^*$ è data dalla somma delle distanze tra tutti i vertici:

$$\sum_{t=1}^{T-1} \|\boldsymbol{Z}(t+1) - \boldsymbol{Z}(t)\|_1.$$

Considerando ogni altro path che visita i nodi nell'ordine: <$Z(\pi_1)$,$Z(\pi_2)$,...,$Z(\pi_T)$>. Sia $\pi_{i^*}$=T, e per tutte le i tali che $1 \leq i < i^*$, se $\pi_{i^*}$=t, allora si definisce $t^+ = \pi_{i+1}$; e inoltre per tutti i valori di i che appartengono all'intervallo $(i^*, T], se\ \pi_i$=T allora si definisce $t^+ = \pi_{i-1}$, e t ∈ {1,2,...,T−1},

Nella pratica si utilizza un'approssimazione dell'algoritmo TSP che è stato dimostrato essere utile nell'analisi dell'espressione genica[6].

Questo algoritmo prima costruisce un albero di *clustering* gerarchico a partire dai dati di espressione misurati e quindi trova l'ordinamento ottimale (in un tempo O( $T^3$) dove T è il numero di campioni) per le foglie dell'albero binario risultante (le foglie dell'albero corrispondono ai dati di espressione misurati).

## 1.2 Ordinamento probabilistico

Fino ad ora si è assunto che le misurazioni di espressione non fossero rumorose e che i profili di espressione dinamici, che nel seguito saranno indicati come "curve di consenso", possano essere ricostruiti senza errore a partire dai dati. In realtà ci sono diversi tipi di rumore che possono far sì che i livelli osservati differiscano dal modello presentato nei paragrafi precedenti, come il rumore sperimentale e la variabilità biologica.

Per considerare anche questi fattori, gli autori del lavoro fornisce un'evoluzione dell'ordinamento ottenuto tramite risoluzione del TSP utilizzando una variante del modello paziente-gene[7]. Il modello utilizza due livelli per rappresentare i profili di espressione: il livello del gene e quello del paziente. Il *gene level* rappresenta il profilo di espressione del gene in risposta al trattamento o alla malattia studiati. Ogni gene viene rappresentato tramite una curva continua, utilizzando le SPLINE (metodo di interpolazione che verrà ripreso nel Capitolo 4); le misurazioni in uno specifico istante temporale corrispondono a campioni rumorosi della curva all'istante specifico.

Il *patient level* ha a che fare con le caratteristiche di tali geni per gli specifici individui. Queste caratteristiche per il paziente vengono parametrizzate da un istante temporale $t_i$ (che rappresenta l'istante di campionamento rispetto alla durata totale). Inoltre, i geni del paziente seguono una mistura di distribuzioni. Una mistura di distribuzioni (*mixture model*) è una variabile casuale la cui funzione di probabilità (nel caso di una variabile

casuale discreta) o funzione di densità di probabilità (nel caso di una variabile casuale continua) è data da una media ponderata di funzioni di probabilità o densità di altre variabili casuali.

Il modello paziente-gene introduce due tipi di parametri:

1) Parametri associati alla rappresentazione continua dei livelli di espressione genica (*gene level*). Questa rappresentazione permette al metodo di gestire dati campionati non uniformemente, situazione tipica per applicazioni cliniche e biologiche.

   Tali parametri includono i punti di controllo delle SPLINE per ogni gene, che vengono denotati con $C_G$ e la varianza del rumore per i livelli di espressione dei geni, denotata con $\sigma^2$.

2) Parametri associati al *patient level*. Includono gli istanti temporali per ogni paziente $t_i$ e un'altra varianza associata alla variabilità biologica dei geni nei soggetti $\beta^2$.

   Dal momento che non si dispone di un buon modello a priori per la variabilità dei geni nei pazienti, $\beta$ sarà verosimilmente maggiore di $\sigma$.

In aggiunta a tali parametri, si associa una variabile random binaria ad ogni gene di ogni paziente $z_{i,g}$. Questa assume valore 1 se il livello di espressione del gene g nel paziente i al tempo $t_i$ è in accordo con il valore assunto dalla curva di consenso in tale istante, mentre assume valore 0 se il livello del gene g non è in accordo con la curva di consenso.

Usando questi parametri e queste variabili, la likelihood dei set di dati di espressione, dato il modello M, si può esprimere come:

$$L(D|M) = \prod_i \prod_g P(D_{i,g} \mid C_g, t_i, \sigma^2)^{Z_{i,g}} P(D_{i,g} \mid \beta^2)^{1-z_{i,g}}, \qquad \text{Eq.(1.1)}$$

dove $D_{i,g}$ è il valore misurato per il gene g nel paziente i. Entrambe le densità sono rappresentate con una gaussiana.

Pertanto la log likelihood dei dati è data da:

$$LL(D|M) = \sum_i \sum_g z_{i,g} \left[\frac{(D_{i,g} - S(t_i)C_g)^2}{2\sigma^2} + log\frac{1}{2\pi\sigma}\right] + (1 - z_{i,g}) \left[\frac{D_{i,g}^2}{2\beta^2} + log\frac{1}{2\pi\beta}\right],$$

Eq.(1.2).

Dove S($t_i$) è un vettore dei coefficienti delle splines valutati al tempo $t_i$.

I parametri del *mixture model* possono essere "appresi" utilizzando un algoritmo di minimizzazione dell'aspettazione (EM) (vedi in Appendice "Algoritmo Expectation-Minimization(EM)"). Si inizia con un'ipotesi iniziale sui valori dei parametri ($C_g$, $t_i$, β, $\sigma$). Al passo E, si usano tali valori per ottenere il valore atteso di $z_{i,g}$ per tutti i geni:

$$z_{i,g} = \frac{p(D_{i,g}|t_i, C_g, \sigma^2)\alpha}{p(D_{i,g}|t_i, C_g, \sigma^2)\alpha + p(D_{i,g}|\beta^2)(1-\alpha)},$$

Eq.(1.3).

Dove α è una stima a priori del numero di geni differenzialmente espressi.

Al passo M si utilizza il valore atteso di $z_{i,g}$ per calcolare le nuove stime dei parametri:

$$\sigma^2 = \frac{\sum_i \sum_g z_{i,g}(D_{i,g} - S(t_i^*)C_g)^2}{\sum_i \sum_g z_{i,g}} \quad , \quad \text{Eq.(1.4)}$$

$$\beta^2 = \frac{\sum_i \sum_g (1-z_{i,g}) D^2{}_{i,g}}{\sum_i \sum_g (1-z_{i,g})} \quad , \quad \text{Eq.(1.5)}$$

Dove l'apice * identifica i valori dei parametri ottenuti al passo M.

I punti di controllo delle spline $C_g$ si ottengono utilizzando la funzione di matlab "spap2". Questa funzione prende come input un insieme di valori pesati e i loro corrispondenti istanti temporali e fornisce in output i punti di controllo delle SPLINE per le SPLINE approssimate che minimizzano l'errore quadrato medio. $t_i$ viene calcolato utilizzando un algoritmo line-search: si campiona ogni curva SPLINE a frequenza di campionamento elevata e, per ogni array, si identifica l'istante temporale che minimizza la somma dei quadrati dello scarto (RSS). Questo istante temporale viene dunque assegnato all'array. Va notato che questo in linea teorica può far sì che due array vengono assegnati allo stesso istante temporale. Nonostante ciò accada raramente

(dal momento che si campiona a una frequenza molto più alta rispetto al numero di soggetti, quindi array), l'algoritmo comunque non ne risente, dal momento che nulla gli vieta di utilizzare due campioni relativi allo stesso istante temporale.

Dal momento che l'algoritmo EM converge solo ad un minimo locale, i valori iniziali assegnati ai parametri giocano un ruolo fondamentale nella ricerca di una buona soluzione. Nel caso del metodo proposto dallo studio si utilizza l'ordinamento dato dal metodo di approssimazione TSP per determinare i valori iniziali di $t_i$, distanziando in maniera uniforme gli array così ordinati.

## 1.3 Performance del metodo

**Dati simulati**

Per mettere alla prova la capacità dell'algoritmo sviluppato di determinare accuratamente l'ordinamento dei dati variando il livello di rumore sperimentale e con differenze di background, nello studio di riferimento sono stati generati valori di espressione simulati. Per ognuno degli n geni si sono selezionati in modo random uno *starting point* e un *ending point* compresi tra 0 e $4\pi$. Si è quindi creata la sinusoide delimitata da tali punti ottenendo una curva continua per ogni gene. Quindi ai diversi geni così generati è stato concesso di cambiare direzione al massimo 4 volte. Ogni curva è stata campionata uniformemente in modo da ottenere 1000 campioni, si è effettuato un campionamento non uniforme e random a T istanti temporali è si è memorizzato il valore di tutti i geni in tali istanti. Tale vettore di T valori dei geni, campionati in maniera non uniforme, a specifici istanti temporali rappresenta un individuo. Si è poi aggiunto rumore random (a media nulla e standard deviation variabile) ad ogni valore dei geni. Infine sono stati selezionati il 15% dei geni indipendentemente per ogni individuo e si sono settati i loro valori con valori campionati da una curva gaussiana con media nulla e standard deviation differenti (rappresentando i valori di espressione di background).

Gli autori hanno quindi applicato l'algoritmo per ricostruire l'ordinamento e i profili di espressione a partire dai T vettori simulati ottenuti come descritto sopra. In Figura 1.3 si possono osservare il profilo originale e quello ricostruito per T=30 e n=200. Come

si può notare l'algoritmo ricostruisce correttamente l'ordinamento e la curva ritrovata per ogni gene ricalca il profilo originale.

Nello studio si è quindi analizzato come il numero di geni n e di individui T influenzi la bontà del risultato ottenuto con l'algoritmo. Come mostra la Figura 1.4 con 500 geni e 15 pazienti l'algoritmo recupera l'ordine corretto più del 90% delle volte. Invece con un numero molto basso di geni a disposizione l'algoritmo spesso non consente di ritrovare l'ordinamento corretto.

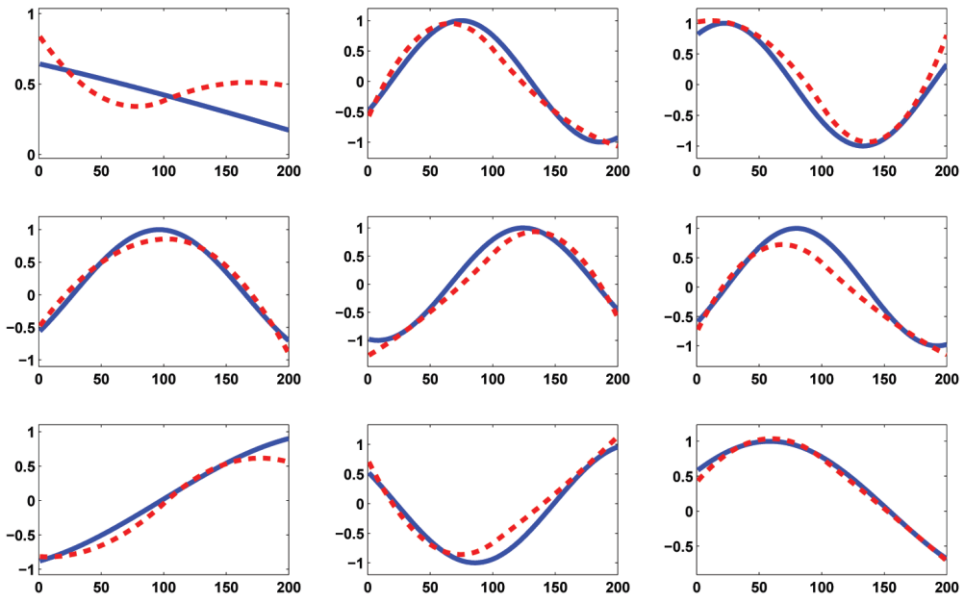


**Figura 1.3.** Confronto tra profili generati (curva continua blu) e ottenuti tramite l'algoritmo (curva tratteggiata rossa). Come si può notare l'algoritmo ricostruisce correttamente l'ordine dei campioni ed è in grado di ricostruire accuratamente i profili per quasi tutti i geni simulati.

Per studiare la dipendenza dell'ordinamento ricostruito dal numero di campioni, gli autori hanno fissato il numero di geni simulati n =200 e si sono creati vari set di dati variando il numero di campioni temporali (soggetti). L'analisi effettuata ha inoltre proposto un confronto tra la versione probabilistica (spline), quella deterministica (euristica TSP) e il metodo PQ-tree di Magwene et al.[8], che peraltro è l'unico lavoro ad aver trattato una problematica analoga al momento della stesura dello studio qui presentato. Come si può notare in Figura 1.5, il metodo probabilistico che tiene conto della variabilità interindividuale ha performance superiori rispetto agli altri due metodi.

L'errore medio indica che per quasi tutti i dati simulati, l'ordinamento ritrovato è discretamente accurato.

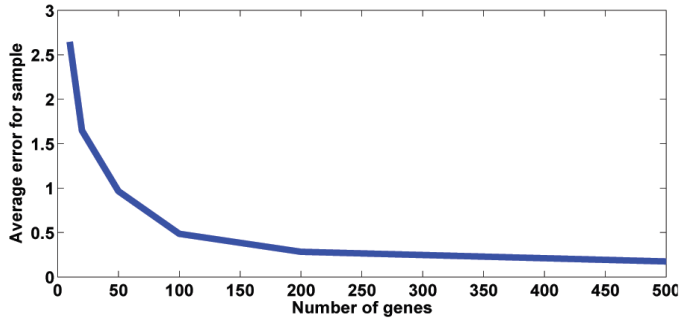
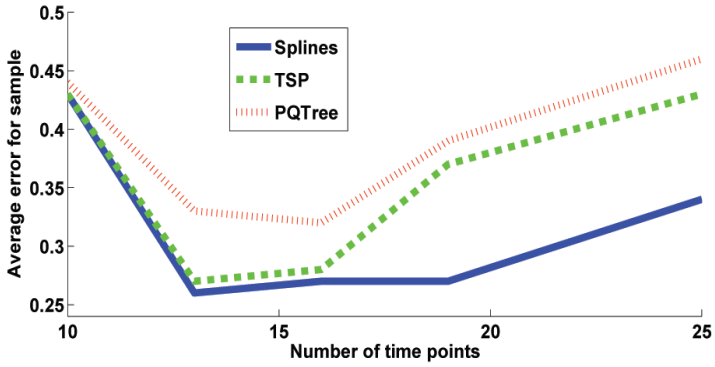



**Figura 1.4**. Dipendenza dell'accuratezza dell'ordinamento ricostruito su dati simulati, dal numero di geni. L'asse y rappresenta la distanza media tra la posizione assegnata nell'ordinamento ricostruito e la posizione corretta. La correttezza dell'ordinamento ottenuto è una funzione del numero di geni. Con uno scarso numero di geni molti campioni non vengono collocati nella corretta posizione. Invece quando il numero di geni raggiunge i 500 una gran parte degli ordinamenti ricostruiti è molto accurata.

**Figura 1.5**. Confronto dell'accurata dei tre metodi al variare del numero di campioni (T), fissato il numero di geni (n=200). Per tutti i metodi l'errore medio è basso anche per un basso numero di campioni (cioè molti campioni sono stati assegnati correttamente alla loro posizione). Il metodo probabilistico basato sulle spline si rivela migliore rispetto al metodo deterministico basato su euristica TSP e rispetto al metodo basato su alberi PQ sui dati simulati.




Per esempio utilizzando 13 campioni temporali, ogni campione è stato assegnato ad una posizione che in media dista 0.3 posti da quella corretta. Il fatto che l'errore aumenti con il numero di istanti temporali è in accordo con le osservazioni teoriche. Il picco iniziale dell'errore (passando da 10 a 13 istanti temporali) è risultato della violazione delle assunzioni su cui si basa il modello adottato, per il quale si assumono piccole variazioni tra campioni temporali consecutivi nell'ordinamento. Se le serie temporali vengono campionate con una frequenza molto bassa questa assunzione non è soddisfatta e si generano errori moto elevati.

Si può notare anche che gli ordinamenti ottenuti con l'euristica TSP nonostante siano buoni, abbiano accuratezza minore rispetto a quanto si ottiene con il modello probabilistico delle spline.

Nello studio considerato si è ripetuta l'analisi utilizzando curve ottenute da polinomi di vario grado. Analogamente a quanto fatto con le curve sinusoidali, si sono selezionati casualmente i coefficienti dei polinomi e poi si sono valutati in punti a caso tra 0 e 1. Il risultato ottenuto si è visto essere simile a quanto ottenuto in precedenza con le curve sinusoidali (vedi Figura 1.6). Come si era osservato in precedenza nella Figura 1.5, il metodo spline è migliore anche in questo caso rispetto sia TSP che PQ-tree, in alcuni casi del 50%.

In conclusione, gli autori hanno notato che l'ordinamento iniziale ottenuto con euristica TSP solitamente raggiunge uno score migliore (più basso) rispetto all'ordinamento corretto (dove per score si intende la somma delle differenze tra vettori di campioni temporali consecutivi nell'ordinamento). Perciò, un diverso algoritmo di approssimazione TSP non avrebbe risolto del tutto questo problema.

L'esistenza di vari tipi di rumore, cosa della quale il modello paziente – gene tiene conto, è la ragione principale degli errori locali commessi dalla soluzione TSP.

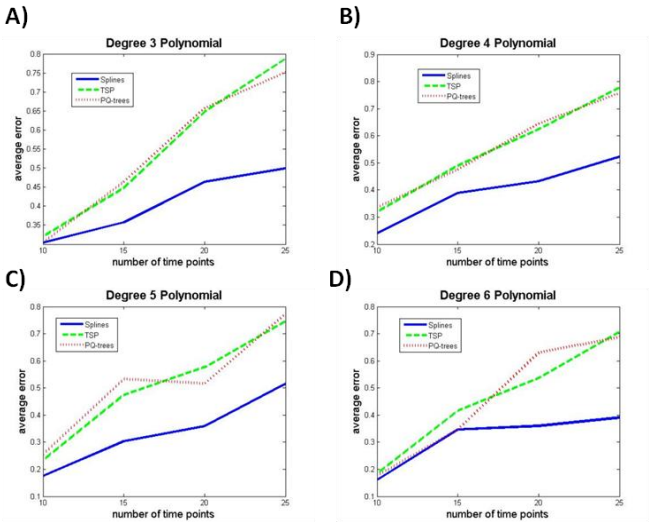



**Figura 1.6**. Confronto dei tre metodi al variare del numero di campioni (T) tenendo fisso il numero di geni (n=200), effettuato su geni simulati generati da curve polinomiali. Si può notare che i risultati concordano con quanto visto in Figura 1.5 (campioni generati da sinusoidi).

**Data set di serie temporali di livelli di espressione nel lievito e nell'uomo**

Per testare l'algoritmo su dati reali il cui ordine sia noto, gli autori dello studio hanno usato data set di serie temporali di espressione provenienti dal lievito e dall'uomo. Per il lievito, sono stati utilizzati 5 serie temporali di dati di espressione tra quelli ottenuti da test di tipo stress-response ricavati da Gash et Al.[9]. Va notato che, come accade per la maggior parte dei data set di questo tipo, questi insiemi di dati contengono molti meno campioni rispetto ai dati relativi a esperimenti effettuati su pazienti colpiti da cancro, e ciò rende più difficile il test.

La Tabella 1.1 presenta i risultati del test. Come si può notare, per tre dei 5 data set l'algoritmo è stato in grado di trovare l'ordinamento corretto. Per gli altri due l'algoritmo ha ritrovato due blocchi il cui ordine interno è corretto e connessi tramite le estremità opposte (campione 1 con 8 o 5). In questo caso la ragione risiede nelle condizioni studiate, ossia esperimenti di tipo *stress – response – recovery*. Nell'esperimento *heat shock* il campione 8 assomiglia infatti al campione non perturbato 1 (precedente al manifestarsi degli effetti della perturbazione) e ciò conduce all'ordinamento ritrovato. Per quanto riguarda l'esperimento di deplezione dell'azoto i primi 4 campioni manifestano piccole variazioni rispetto al livello basale, e l'effetto inizia solo in apparenza al quinto istante temporale. Ciò potrebbe considerarsi un problema del metodo di ordinamento proposto, ma va considerato che si tratta di situazioni sperimentali che non si ripropongono frequentemente e anzi sono pressoché da escludersi in esperimenti su dati di espressione statici, incluso il caso di esperimenti sul decorso di una patologia. Tali tipi di data set infatti contengono campioni raccolti da pazienti per i quali è stata di recente diagnosticata la patologia e nei quali la malattia sta progredendo.

| Condition | Num. time points | Sampling rate (min.) | Recovered order |
|---|---|---|---|
| DTT | 8 | 5, 15, 30, 45, 60, 90, 120, 180 | 1, 2, 3, 4, 5, 6, 7, 8 |
| Amino acid starvation | 5 | 30, 60, 120, 240, 360 | 1, 2, 3, 4, 5 |
| Diauxic shift | 7 | 0, 570, 690, 810, 930, 1110, 1230 | 1, 2, 3, 4, 5, 6, 7 |
| Nitrogen depletion | 10 | 0.5h, 1, 2, 4, 8, 12, 24, 48, 72, 120 | 4, 3, 2, 1, 5, 6, 7, 8, 9, 10 |
| Heat shock | 8 | 5, 10, 15, 20, 30, 40, 60, 80 | 1, 8, 7, 6, 5, 4, 3, 2 |

**Tabella 1.1.** Ordinamento ricostruito per data set dinamici ottenuti da esperimenti sul lievito.

Nello studio si sono quindi analizzati 4 data set di espressione genica umana ottenuti da Baldwin et Al.[10], al fine di studiare la risposta di cellule umane ad un'infezione da parte di un batterio patogeno. Come parte della loro analisi gli autori hanno osservato 4 serie temporali di espressione, 2 per una versione wild-type dell'agente patogeno e 2 per una versione mutante. Tutte le serie sono state ottenute tramite 6 esperimenti campionati a 0, 30, 60, 120, 240 e 480 minuti. Nella tabella 1.2 si può osservare l'ordinamento ricostruito.

**Tabella 1.2.** Ordinamento ricostruito per dati di serie temporali ottenuti da esperimenti su cellule umane.

| Condition | Recovered order: Our model | Recovered order: TSP | Recovered order: PQ-trees |
|---|---|---|---|
| Wild type 1 (WT1) | 1, 2, 3, 4, 5, 6 | 6, 1, 2, 3, 4, 5 | 1, 2, 3, 4, 5, 6 |
| Wild type 2 (WT2) | 3, 2 ,1, 5 ,4 ,6 | 3, 2 ,1, 5 ,4 ,6 | 3, 2 ,1, 5 ,4 ,6 |
| Mutant 1 (M1) | 1, 3, 2, 6, 5, 4 | 1, 3, 2, 6, 5, 4 | 1, 3, 5, 6, 2 ,4 |
| Mutant 2 (M2) | 1, 2, 3, 4, 6, 5 | 1, 2, 3, 4, 6, 5 | 1, 2, 3, 4, 6, 5 |

Come si può osservare il risultato ottenuto in due casi (WT1 e M2) è corretto o quasi corretto. Per WT2 e M1 si ha una separazione corretta tra prima e seconda metà temporale però l'ordine interno non è esatto. Va notato che il numero di istanti temporali in esperimenti di serie temporali è veramente esiguo e che per esperimenti statici il numero di campioni è molto più elevato.

Un'altra cosa da notare è che, mentre per tre dei quattro data set, l'ordinamento ritrovato è il medesimo per i due metodi proposti nello studio considerato in questo capitolo e per il metodo basato su PQ-tree, per il quarto data set (M1) il metodo probabilistico conduce invece a risultati migliori rispetto ad entrambi gli altri. Infatti si

ha una separazione corretta delle due metà dell'esperimento cosa che non accade con il metodo basato su PQ-tree.

**Dati relativi al cancro**

Per testare la capacità dell'algoritmo di ordinare dati di espressione statici raccolti da pazienti affetti da cancro si utilizza un data set fornito da Nutt et al.[11] relativo a pazienti affetti da glioma maligno (vedi in Appendice B."Il glioma"), il tumore cerebrale più comune (nel loro articolo i dati vengono usati al fine di ottenere un metodo di classificazione del glioma maligno). Sono stati misurati approssimativamente 12000 geni in un insieme di 50 gliomi: 28 glioblastomi e 22 oligodendrogliomi anaplastici, questo insieme è stato diviso in 2 set: 1 training set, contenente 21 tumori con istologia classica (in particolare 14 glioblastomi e 7 oligodendrogliomi) e un test set, contenente 29 tumori con istologia non classica, quindi più difficili da diagnosticare (nello specifico 14 glioblastomi e 15 oligodendrogliomi). Gli autori dello studio dal quale sono stati raccolti i dati, hanno utilizzato il metodo KNN (*k nearest neighbor*) settando k=3 e un metodo di *feature selection* che è risultato selezionare 20 geni.


**Figura 1.7.** Analisi di dati di espressione relativi a cancro. (a) tempo di sopravvivenza (in giorni) relativo all'ordinamento ottenuto dall'algoritmo. Da notare che alcuni pazienti erano in vita nel momento in cui i dati sono stati analizzati. Molti pazienti in vita sono stati assegnati alla metà destra dell'ordinamento ottenuto dall'algoritmo e i pazienti da questa parte vivono più a lungo. Una possibile spiegazione è che a essi sia stata diagnosticata la malattia prima e ciò indica che l'ordinamento è ragionevole. Questo plot rappresenta l'ordinamento ricostruito per l'intero test set dal momento che l'ordinamento determinato dall'algoritmo è reversibile nel tempo (l'algoritmo non può discernere istante iniziale e finale), le informazioni relative alla sopravvivenza tornano utili. Utilizzando questa informazione possiamo concludere che il tempo di inizio si trova a destra mentre quello finale a sinistra.

Per prima cosa gli autori hanno usato l'algoritmo per determinare l'ordinamento per il training set (per ognuno dei sottotipi di glioma e per l'unione dei due). L'ordinamento ottenuto dall'algoritmo si è visto essere in accordo con il tempo di sopravvivenza successivo al momento in cui sono stati raccolti i campioni. La Figura 1.7 rappresenta la durata di sopravvivenza dei pazienti dal momento della raccolta dei livelli di espressione in poi in funzione della collocazione del soggetto nell'ordinamento fornito dall'algoritmo. La durata media di sopravvivenza per la metà destra dei campioni (riferendoci all'ordinamento) è di 1137 giorni mentre la metà sinistra di 359.

Seguendo l'ordinamento trovato per entrambi i data set ( training e test) gli autori hanno usato una rappresentazione continua derivata dall'algoritmo per implementare un semplice classificatore. Sono state campionate uniformemente ad alta frequenza le curve per ogni gene, ottenendo un insieme di vettori per ogni data set denotati con $V_1$ e $V_2$. Ogni vettore $v_1^t \in V_1$contiene i valori di tutti i geni relativi all'istante temporale t e al set di dati 1 (test e training). Dato un nuovo campione, si è calcolata la distanza euclidea tra il valore dei geni del nuovo campione e ogni vettore in $V_1$ e $V_2$ e si è assegnato il campione alla classe del vettore con la distanza euclidea minore.

Sono dunque stati confrontati i risultati ottenuti dal classificatore costruito con il metodo realizzato in questo studio con il classificatore KNN suggerito nell'articolo originale e con un classificatore basato su SVM (*support vector machine*). Per il classificatore KNN sono stati usati entrambi i set di *neighbors* (3) e *features* (100) riportati nell'articolo originale, come anche l'insieme ottimo di *neighbors* (1) e *features* (100) ottenuto tramite una *leave one out cross-validation*. L'articolo originale non conteneva l'implementazione così nello studio [1] si è implementato *ex-novo* il metodo, incluso l'algoritmo di *features selection*[12]. Per quanto riguarda il metodo descritto in questo studio il miglior risultato sul training set si è ottenuto utilizzando 750 geni. Le *features* sono state selezionate scegliendo un insieme di geni caratterizzate dal più elevato valore dell'integrale al quadrato (quindi in termini assoluti) tra le curve ricostruite per le due classi. Come si può vedere in figura, nonostante uno dei metodi KNN conduca a risultati migliori sulla cross-validation test, questi risultati non confermano al loro bontà se riportati al test set. Invece la performance dell'algoritmo sviluppato, nel test set rispecchia quanto si ottiene anche nel training set e si rivela decisamente superiore ad entrambi i metodi KNN. Anche SVM non raggiunge performance elevate.

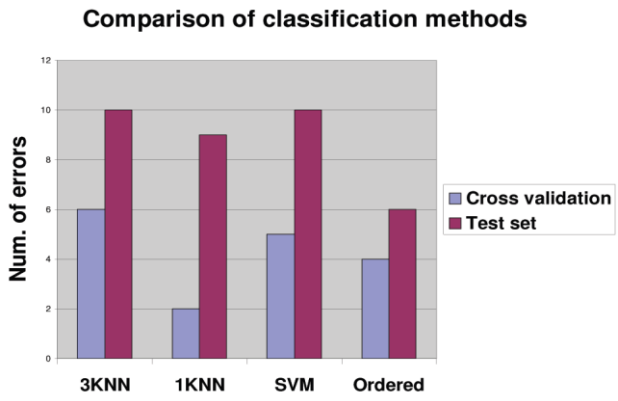



**Figura 1.8.** Classificazione del training e del test set. Si sono confrontati i risultati dell'algoritmo sviluppato in questo studio con tre altri classificatori: due KNN e un SVM. Mentre tutti i classificatori funzionano abbastanza bene sul training set, il metodo ordinato temporalmente funziona decisamente meglio sul test set. La diversa istologia tra traning e test set si ripercuote sugli altri metodi.

La natura del training e del test set (istologie classiche e non classiche) può portare a overfitting con i metodi KNN e SVM. Mentre, grazie alle informazioni temporali, il metodo di ordinamento temporale può funzionare bene anche nei casi in cui il test set sia concentrato ad uno specifico insieme di istanti temporali.

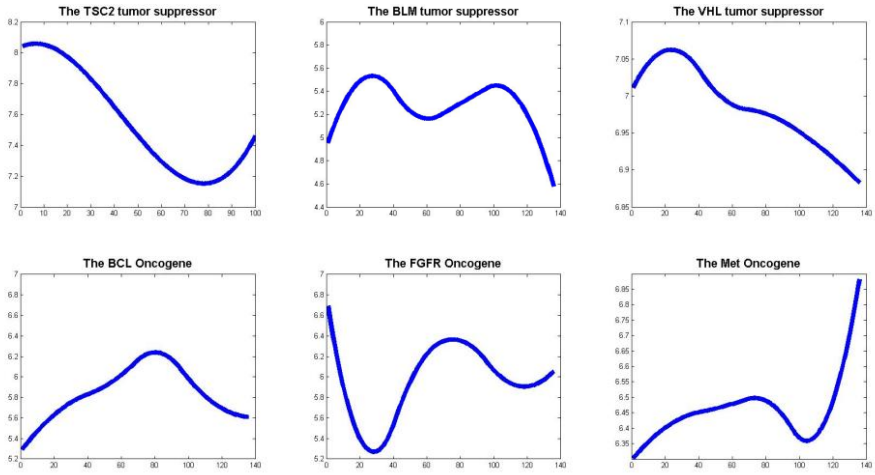
Come passo successivo gli autori hanno utilizzato i profili ottenuti per ogni gene per identificare i geni significativi. Si è assegnato il tempo d'inizio alla parte della curva caratterizzata dal tempo di sopravvivenza medio più alto. Sono stati identificati 140 geni down-regolati e 40 up-regolati.

Si è poi utilizzata la Gene Ontology (GO) per analizzare queste due liste. Per il data set caratterizzato da livello di espressione decrescente (geni down–regolati) la categoria più "arricchita" (*enriched*) si è rivelata essere la matrice extracellulare (P value<$7*10^{-11}$). Le proteine nella matrice extracellulare comunicano con il nucleo cellulare, modificandone la struttura e come conseguenza sono in grado di generare un espressione selettiva dei geni. È stato inoltre recentemente dimostrato[13] che mutazioni in tali geni possono comportare cancro al seno. Quindi la graduale decrescita di espressione di questi geni può spiegare la progressione del cancro. Un'altra categoria che è stata "arricchita" per i geni con espressione decrescente è la "regolazione negativa di proliferazione cellulare" (P value<$7*10^{-4}$ ). Come noto la mancanza di controllo sul ciclo cellulare è una delle caratteristiche peculiari del cancro. Il fatto che il livello di espressione di questi geni sia decrescente può indicare che il sistema via via che la

malattia stia progredendo perda il contributo di alcuni dei suoi geni di controllo fondamentali.

L'insieme dei geni con espressione crescente è stata invece arricchita con diverse categorie della GO. La categoria più arricchita è "lo sviluppo del sistema nervoso" (P value<$3*10^{-4}$), la quale contiene geni che ci solitamente sono up-regolati durante la generazione di cellule cerebrali. Un'altra categoria rivelatasi importante è la "differenziazione cellulare" (P value<$5*10^{-3}$) la quale contiene geni associati alla crescita e divisione cellulare, incluso il fattore di crescita dei fibroblasti 9 (FGF9), che recentemente è stato dimostrato giocare un ruolo fondamentale nella crescita del tumore[14].

Come ultima analisi si è osservata la curva ricostruita per alcuni degli oncogeni noti e per alcuni *tumor-suppressors, utilizzando* OMIM[15]. Molti dei geni considerati hanno mostrato un pattern di espressione coerente con quanto ci si aspettava. Per esempio, l'espressione del gene TSC2, un gene *tumor-suppressor,* diminuisce con la progressione della malattia.




3

**Figura 1.9**. Si sono osservate le curve ricostruite relative ad alcuni geni oncogeni e tumor-suppressors noti. Utilizzando il data-base OMIM per selezionare geni che presenti nel data set che mostrassero una variazione di espressione abbastanza significativa. I pattern di espressione di molti di tali geni evidenziano il comportamento atteso. Per esempio, l'espressione del TSC2, un gene tumor-suppressor, diminuisce con il progredire della malattia. Nei grafici è possibile osservare il profilo di espressione ricostruito per alcuni geni noti.

## 1.4 Vantaggi e limiti del metodo

Nello studio analizzato, si è mostrato come sotto opportune assunzioni sulle dinamiche dei geni presi individualmente, sia possibile ricostruire un ordinamento temporale a partire da dati statici di espressione.

Il modello visto assume che durante la progressione della malattia i geni non varino spesso la direzione della traiettoria. Tale assunzione è decisamente verosimile e tollerabile per gran parte dei data set di espressione genica. Una volta descritto il modello per un singolo gene gli autori hanno dimostrato che il metodo risolutivo TSP porta a conclusioni molto buone e permette di trovare ordinamenti soddisfacenti. Quindi si proceduto estendendo il metodo in modo da poter trattare adeguatamente dati più realistici, ossia dati rumorosi; cioè si è utilizzato l'ordinamento trovato tramite TSP come ipotesi iniziale per un successivo algoritmo in grado di ricostruire curve continue probabilistiche per rappresentare il profilo di espressione dei geni. Questo metodo si è rivelato molto buono nel ricostruire l'ordine di serie di dati di espressione simulati, e ha mostrato buone performance anche con dati reali, ordinando data set provenienti da cellule di lievito e cellule umane.

Il metodo proposto fornisce prestazioni più elevate con data set caratterizzati da un considerevole, ma non eccessivo numero di soggetti (istanti temporali). Per un numero di campioni molto esiguo alcune delle assunzioni fatte vengono meno, in particolare nel caso di campionamento molto rado le assunzioni sulla costanza della direzione della traiettoria cadono.

Gli autori hanno inoltre testato l'algoritmo su dati provenienti da campioni cellulari umani relativi a pazienti affetti da varie tipologie di glioma (uno dei più diffusi tumori cerebrali). Valutare la correttezza dell'ordinamento ritrovato in tal caso è chiaramente molto difficile, ma per avere un'idea gli autori hanno fatto un'analisi dei tempi di sopravvivenza dei pazienti utilizzati per la raccolta dei dati e gli hanno confrontati con l'ordinamento ritrovato dall'algoritmo. Come messo in evidenza l'ordinamento ritrovato concorda con i tempi di sopravvivenza e dunque si può pensare che ben rifletta lo stadio dell'evoluzione biologica della patologia.

Utilizzando i profili ricavati per ogni gene si è dunque effettuata una classificazione per i due tipi di glioma considerati, ottenendo risultati migliori rispetto a quanto ottenuto con metodi suggeriti in studi precedenti (SVM e KNN).

Il vantaggio più rilevante relativo al metodo proposto se confrontato con gli altri due è la capacità di distinguere, per ogni tipologia di cancro, i geni che esprimono differenze dovute al rumore e geni che evidenziano differenze di espressione dovute al momento dello stadio evolutivo della patologia al quale appartengono. Mentre i geni che esprimono solo rumore non sono rilevanti ai fini della classificazione, il secondo gruppo di geni può invece essere di grande aiuto. L'abilità posseduta dall'algoritmo di identificare questi geni può essere d'aiuto anche per determinare una relazione causale con la progressione della malattia. Infatti si è verificato che, tramite GO, una buona parte dei geni che hanno evidenziato un comportamento significativo (up-down regolati) sono relazionati alla progressione dalla malattia.

Va notato che l'ordinamento ottenuto tramite il metodo sviluppato è reversibile e ciò consente di effettuare verifiche e ulteriori messe a punto se in possesso di ulteriori informazioni a priori. Per esempio le informazioni riguardanti i tempi di sopravvivenza hanno permesso  individuare il punto di start più appropriato per i dati relativi al cancro (ciò è utile dal momento che l'algoritmo tende a suddividere in blocchi).

# Appendici al Capitolo 1

## A. The Travelling Salesman Problem (TSP):

Il problema del commesso viaggiatore, o TSP (dall'inglese Travelling Salesman Problem) è un problema di teoria dei grafi, uno dei casi di studio tipici dell'informatica teorica e della teoria della complessità.

Il nome nasce dalla sua più tipica rappresentazione: data una rete di città, connesse tramite delle strade, trovare il percorso di minore lunghezza che un commesso viaggiatore deve seguire per visitare tutte le città una e una sola volta.

Espresso nei termini della teoria dei grafi è così formulato: dato un grafo completo pesato, trovare il ciclo hamiltoniano con peso minore. La rete di città può essere rappresentata come un grafo in cui le città sono i nodi, le strade gli archi e le distanze i pesi sugli archi.

Il problema è di considerevole importanza pratica, al di là delle ovvie applicazioni nella logistica e nei trasporti. Non esistono algoritmi efficienti per la risoluzione del TSP, l'unico metodo di risoluzione è rappresentato dall'enumerazione totale, ovvero nell'elaborazione di tutti i possibili cammini sul grafo per la successiva scelta di quello migliore. Tuttavia, la complessità dell'operazione la rende impraticabile per grafi di dimensioni comuni nei problemi reali: in un grafo di n nodi, bisognerà calcolare, nel caso peggiore in cui ogni nodo è connesso con tutti gli altri, n! (n fattoriale) possibili cammini, il che implica una complessità esponenziale (in base all'approssimazione di Stirling). Il TSP rappresenta inoltre un esempio di problema di programmazione lineare intera nel quale risulta pressoché inattuabile ottenere valutazioni approssimate tramite la tecnica del rilassamento continuo poiché il problema di programmazione lineare risultante si troverebbe ad avere un numero di vincoli che cresce in modo esponenziale con i nodi, rendendo così intrattabili le matrici associate ad esso. Per scopi pratici, viene in generale usato il metodo del “simulated annealing”.


**Figura A.1.10.** Un cammino ottimo TSP attraverso la 15 città maggiori della Germania.
È il più breve dei 43 589 145 600 cammini possibili che visitano le città una sola volta.

## B. Il Glioma:

I tumori primari del sistema nervoso centrale (SNC) comprendono un variegato insieme di entità patologiche ciascuna con una sua distinta storia naturale.

Per il fatto che i tumori della glia costituiscono da soli quasi il 40% di tutti i tumori del SNC in letteratura si è soliti operare una distinzione tra tumori gliali (o gliomi) e tumori non gliali.Le cellule della glia, dette anche cellule gliali, sono cellule che, assieme ai neuroni, costituiscono il sistema nervoso. Hanno funzione nutritiva e di sostegno per i neuroni, assicurano l'isolamento dei tessuti nervosi e la protezione da corpi estranei in caso di lesioni.

I gliomi più comuni sono gli astrocitomi (che originano dalle cellule astrocitiche della glia), gli oligodendrogliomi (dalle cellule oligodendrogliali) e gli ependimomi (dalle cellule ependimali).

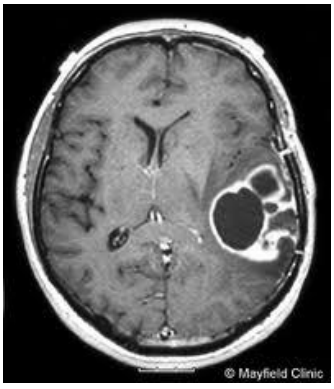



**Figura B.1.11.** Immagine MRI di un glioma nel lobo parietale.

## C. Algoritmo Expectation-Maximisation (EM)[16]:

Lo scopo dell'algoritmo EM è quello di aumentare, e possibilmente di massimizzare, la likelihood dei parametri di un modello probabilistico M rispetto ad un insieme di dati *s*, risultati di un processo stocastico che coinvolge un processo non noto. Indicando con $\theta^0$ i parametri correnti del modello, lo scopo è dunque quello di ottenere un nuovo insieme di parametri $\theta$ tali che:

$$\log P(s|\theta, M) - \log P(s|\,\theta^0, M) \geq 0$$

Introducendo le variabili nascoste, si ha:

$$P(s|\theta, M) = \frac{P\,(s, \pi|\theta, M)}{P\,(\pi|s, \theta, M)}$$

Quindi, passando ai logaritmi:

$$\log P(s|\theta, M) = \log P(s, \pi|\,\theta, M) - \log\, P(\pi|s, \theta, M)$$

Moltiplicando per la distribuzione di probabilità della variabile nascosta dati i parametri attuali, $P(\pi|s, \theta^0, M)$, e sommando su tutti i valori che può assumere la variabile nascosta si ottiene:

$$\log P(s|\theta, M) = \sum_\pi P(\pi\,|s, \theta^0, M)(logP(s, \pi|\theta, M) - logP(\pi|s, \theta, M) \qquad (4)$$

Viene definita una funzione ausiliaria $Q(\theta|\theta^0)$, come valore di aspettazione del logaritmo della probabilità congiunta di *s* e *p* sui possibili valori della variabile nascosta:

$$Q(\theta|\theta^0) = \sum_\pi P(\pi\,|s, \theta^0, M)(logP(s, \pi|\theta, M)$$

L'espressione da rendere massima diviene:

$$\log P(s|\theta, M) - \log P(s|\theta^0, M) =$$

$$= Q(\theta|\theta^0) - Q(\theta^0|\theta^0) - \sum_\pi P(\pi\,|s, \theta^0, M) log \frac{P\,(\pi|s, \theta, M)}{P\,(\pi|s, \theta^0, M)}$$

Il terzo termine del secondo membro di questa uguaglianza è l'entropia relativa delle distribuzioni $P(\pi|s,\theta,M)$ e $P(\pi|s,\theta^0,M)$ che, per quanto visto nella sezione precedente, è sempre positiva. Ne consegue che:

$$\log P(s|\theta,M) - \log P(s|\theta^0,M) \geq Q(\theta|\theta^0) - Q(\theta^0|\theta^0)$$

Questa disuguaglianza è il nucleo dell'algoritmo EM. Infatti se si può calcolare un insieme di parametri $\theta^0$che renda positiva la differenza delle funzioni ausiliarie, questo incrementerà la likelihood del modello rispetto ai dati. In particolare, l'obiettivo è trovare i valori $\theta^{MAX}$ che massimizzano tale differenza, ovvero:

$$\theta^{MAX} = \text{argma}x_\theta(Q(\theta|\theta^0)) ;$$

L'algoritmo EM si compone dunque di due passi:

- Calcolo del valore di aspettazione $Q(\theta|\theta^0)$ a partire dai parametri del modello attuale.
- Massimizzazione di $Q(\theta|\theta^0)$nelle variabili $\theta$ e aggiornamento del modello.

A partire da un'ipotesi iniziale sui parametri del modello questi due passi vengono applicati iterativamente fino al raggiungimento della convergenza quando l'aggiornamento dei parametri non incrementa più la likelihood. L'algoritmo non assicura il raggiungimento della likelihood massima globalmente, ma solo il suo incremento ad ogni successiva applicazione e la convergenza ad un massimo locale. Inoltre a volte non e' possibile effettuare in modo esatto il passo di massimizzazione, o almeno non in modo efficiente e computazionalmente poco dispendioso.

## D. Algoritmo Line-Search [17][18][19] [20]:

Nell'ottimizzazione non vincolata la strategia *line search* è uno dei due approcci iterativi di base per ricercare un minimo locale $\mathbf{x}^*$ di una funzione obiettivo *f*: $\mathbb{R}^n \rightarrow \mathbb{R}$. L'altro approccio è chiamato *trust region*.

L'approccio *line search* prima sceglie una direzione discendente $d^k$, a partire dal punto corrente $x^k$, e cerca lungo la direzione $d^k$ un nuovo punto con un valore della funzione obiettivo più basso:

$$x^{k+1} = x^k + \alpha d^k, \quad \text{con f}(x^{k+1}) < \text{f}(x^k)$$

dove il valore di $\alpha$, detto lunghezza del passo, viene determinato risolvendo il seguente problema:

$$min_\alpha = f(x^k + d^k)$$

Il successo del metodo risiede nella scelta della direzione di discesa e della lunghezza del passo e lo scopo principale é di trovare il minimo con il minor numero di valutazioni della funzione f, del gradiente $\nabla$f e dell'hessiano $\nabla^2$f.

# Capitolo 2

# Razionale e obiettivo dello studio

Lo studio esposto nel Capitolo 1 ha un potenziale estremamente elevato dal momento che la possibilità di ricavare profili dinamici da dati di espressione genica statici permetterebbe di poter effettuare analisi usualmente rivolte a serie temporali, quindi con un contenuto informativo decisamente più elevato, su un numero di data set notevolmente maggiore. Inoltre i data set "dinamici" così ricreati sarebbero molto più estesi dei consueti data set da serie temporali di espressione, a ulteriore vantaggio del contributo informativo. Come visto alla fine del Capitolo 1, una volta applicato il metodo e ottenuti i profili dinamici le possibili ulteriori analisi sono molteplici, e di tale aspetto non ci occuperemo in questa sede.

In questo studio ci preoccuperemo di fornire una validazione statisticamente più significativa del metodo proposto nel lavoro di Anupam Gupta e Z. Bar-Joseph [1], proporre ulteriori elaborazioni finalizzate a migliorarne le prestazioni e infine di validare tali estensioni del metodo su dati di espressione genica simulati e reali.

Nel lavoro di riferimento i data set simulati realizzati per validare il metodo non contemplano la presenza di geni non significativi; ciò è un'assunzione non tollerabile dal momento che la maggior parte dei profili risultanti da esperimenti high-throughput su microarray è invece relativa a geni non coinvolti nel processo biologico considerato. Inoltre nello studio si considerano data set di dimensioni comunque ridotte rispetto ai data set statici che si possono avere a disposizione (in cui si analizzano migliaia di geni e a volte più di cento soggetti). Per queste ragioni il metodo verrà testato su data set simulati estesi contenenti un numero elevato di profili e di soggetti.

Come visto in 1.4 il metodo proposto ha lo svantaggio di fornire molte volte ordinamenti strutturati in blocchi, corretti al loro interno, ma permutati tra loro e a volte con senso di percorrenze invertito rispetto all'ordinamento "vero". Per tale problematica nello studio di riferimento non viene proposta alcuna soluzione, ma data la sua evidente importanza, nello studio corrente verranno proposte e validate due possibili vie risolutive.

La prima soluzione proposta utilizza la tecnica meta-euristica del Bootstrap, cioè una metodologia computazionale basata sul ricampionamento iterativo dei dati al fine di migliore la robustezza del metodo. La seconda si basa su un procedimento *ad-hoc* per la risoluzione del problema della presenza di profili suddivisi in blocchi. È lecito supporre che la derivata di un profilo originariamente regolare, spezzettato in blocchi permutati tra loro e in verso di percorrenza sia diretto che inverso, presenti dei picchi in corrispondenza ai punti di rottura (i campioni estremi dei blocchi). Sfruttando questa considerazione e riportandola dalla scala del singolo profilo, a quella dell'intero data set, sarà così possibile individuare i blocchi, in un secondo momento ricombinarli tra loro (permutandoli e variandone il verso), e infine scegliere la combinazione winner che minimizzi una funzione costo. Un esempio di funzione costo può essere il valor medio dei candidati picchi di ogni possibile permutazione dei blocchi.

Una volta sviluppate le due metodologie (bootstrap e riassetto dei blocchi) si procederà con la loro validazione su un numero molto elevato di data set simulati, in modo da avere un'attendibilità statistica del risultato (positivo o negativo che sia). Nel caso di risultati soddisfacenti sui dati simulati si effettuerà un'applicazione a dati di espressione statici reali ottenuti da pazienti affetti da LLC, confrontando i risultati ottenuti con i valori dei marker prognostici valutati nella consueta pratica clinica.

# Capitolo 3

# Dati

## 3.1 Dati simulati

Per valutare la capacità e le prestazioni delle metodologie di ordinamento proposte in questo lavoro sono stati generati e utilizzati dei dati simulati. Tali dati sono stati generati in modo da ottenere il livello di espressione di 1000 monitorati in 50 soggetti diversi. Come suggerito nel lavoro di Di Camillo-Toffolo[21] sono stati generati 120 geni (sui 1000 geni totali) come differenzialmente espressi, a partire da 6 diversi *pattern* temporali (Figura 3.1) e 880 profili non significativi. I profili 1, 2 e 6 di Figura 2.1 variano tra i valori -1 e +1, mentre i profili 4 e 5 variano tra 0 e 1.

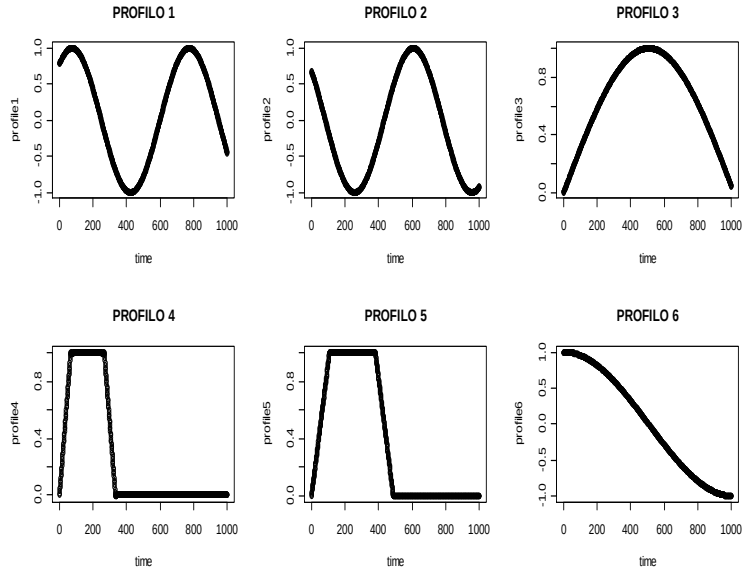



**Figura 3.1.** Pattern temporali usati per generare i 120 geni differenzialmente espressi simulati

Mentre i profili 4 e 5 sono stati ottenuti unendo andamenti lineari, i profili 1, 2, 3 e 6 sono stati generati come descritto in [1] , cioè a partire da andamenti sinusoidali (funzioni seno e coseno) considerati a differenti istanti di inizio e fine (compresi tra 0 e $2\pi$). Si è fatto in modo che ad ognuno dei 6 profili corrispondesse lo stesso numero di campioni (1000, campionamento molto fitto). I geni differenzialmente espressi sono stati generati da questi 6 pattern temporali utilizzando il modello:

$$X_i = k_i * P + q_i + \Sigma \qquad \text{Eq.(3.1)}$$

Dove P = < p(1),...,p(m) > è il vettore di m=1000 valori (campioni) corrispondente al pattern temporale considerato, $k_i$ e $q_i$ sono parametri gene-specifici e $\Sigma$ è una componente di errore additiva. Per ognuno dei 6 pattern sono stati generati 20 geni, dei quali 10 con $k_i$ e $q_i$ campionati da una distribuzione uniforme nell'intervallo $\pm$(0.5, 2) e (-0.5, 0.5) rispettivamente, e 10 geni con $k_i$ e $q_i$ campionati da una distribuzione uniforme nell'intervallo $\pm$ (1, 3) e (-3, 3) rispettivamente. La componente rumorosa additiva è stata ottenuta a partire da una distribuzione gaussiana a media nulla e standard deviation pari a 0.20. In Figura 3.2 è possibile osservare gli andamenti di alcuni tra i profili relativi a geni differenzialmente espressi. Per creare gli 880 geni non significativi si è proceduto generando un profilo costituito da 1000 campioni aventi valore pari a 0 e applicando su questo il modello visto in Eq. (3.1) con i valori visti sopra per i parametri. Quindi si sono uniti in ordine random i 1000 geni in modo da ottenere una matrice avente numero righe = (1000, numero geni) e colonne = (1000, numero campioni).È stata quindi generata una seconda matrice ottenuta campionando casualmente 50 delle 1000 colonne della matrice ottenuta in precedenza, ottenendo così la matrice dei dati di espressione statica relativa a 50 soggetti sintetici. Di tali soggetti si è memorizzato l'ordinamento prima di randomizzare le colonne e generare una matrice di dati di espressione genica con i soggetti ossia le colonne in ordine temporale casuale, ma con ordine reale noto in modo tale da poter valutare la bontà dell'ordinamento ottenuto.

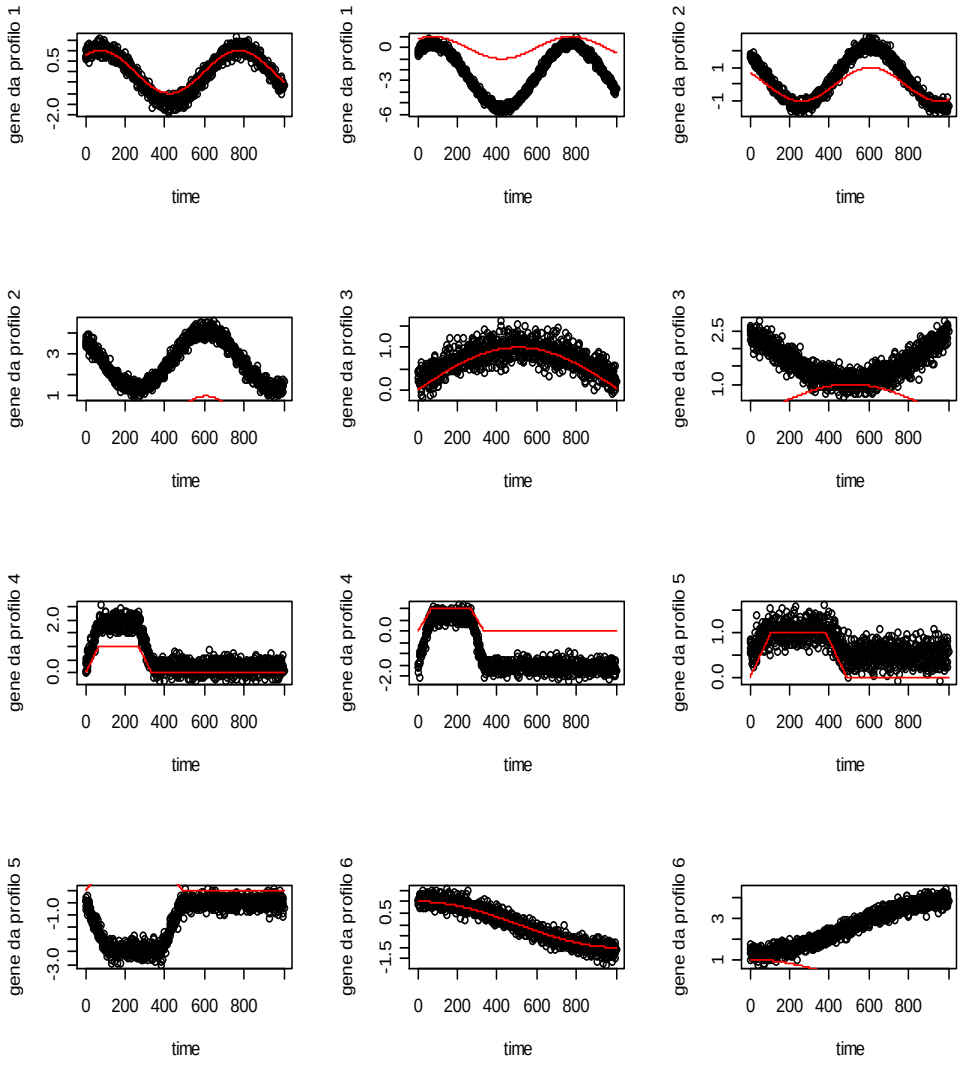



**Figura 3.2.** Profili di geni differenzialmente espressi generati dai pattern di Figura 3.1, secondo il modello descritto in Eq.(3.1).

Per testare la capacità del metodo deterministico di riordinare correttamente data set estesi sono stati generati data set costituiti da 1000 geni simulati, dei quali 120 generati come differenzialmente espressi, e 50 campioni-soggetti. Per avere validità statistica si sono creati 320 data set affetti da livelli di rumore diverso. Per ognuno dei 16 livelli di rumore sono stati realizzati 20 data set indipendenti. In Figure 3.3 è possibile osservare una descrizione completa dei data set sintetici generati.

I. 20 DATA SETS CON SD=0
II. 20 DATA SETS CON SD=0.20
III. 20 DATA SETS CON SD=0.25
IV. 20 DATA SETS CON SD=0.30
V. 20 DATA SETS CON SD=0.35
VI. 20 DATA SETS CON SD=0.40
VII. 20 DATA SETS CON SD=0.45
VIII. 20 DATA SETS CON SD=0.50
IX. 20 DATA SETS CON SD=0.60
X. 20 DATA SETS CON SD=0.70
XI. 20 DATA SETS CON SD=0.80
XII. 20 DATA SETS CON SD=0.90
XIII. 20 DATA SETS CON SD=1
XIV. 20 DATA SETS CON SD=1.20
XV. 20 DATA SETS CON SD=1.40
XVI. 20 DATA SETS CON SD=1.60

**Figura 3.3**. Panoramica dei 320 data set sintetici generati per testare le performance dell'algoritmo deterministico. La standard deviation SD descrive il livello di rumore con il quale sono stati generati i dati.

Per quanto riguarda la validazione del bootstrap sono stati generati 20 data set ognuno composto da 1000 profili di espressione sintetici campionati con B=50 ( B è il numero di campioni) dei quali 880 non differenzialmente espressi e 120 differenzialmente espressi, e si ha quindi reiterato questa procedura 17 volte indipendentemente variando il livello di rumore aggiunto ai dati. Il rumore è stato ottenuto da una distribuzione uniforme con media nulla e standard deviation di 17 valori diversi. In tab. 3.1 si ha un riassunto delle tipologie di dati sintetici generati.

Infine per testare il metodo deterministico "arricchito" con il successivo riordinamento dei blocchi sono stati generati 50 data set simulati come visto in 2.1. Per ogni data set sono state create 9 copie affette da differenti livelli di rumore crescente. Il rumore aggiunto è caratterizzato da *standard deviation* pari rispettivamente a:

$$\text{vett\_SD} = < 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 >.$$

| | Noise ( mean ) | Noise ( standard deviation ) |
|---|---|---|
| Group A ( 20 Data sets ) | 0 | 0 |
| Group B ( 20 Data sets ) | 0 | 0.10 |
| Group C ( 20 Data sets ) | 0 | 0.20 |
| Group D ( 20 Data sets ) | 0 | 0.25 |
| Group E ( 20 Data sets ) | 0 | 0.30 |
| Group F ( 20 Data sets ) | 0 | 0.35 |
| Group G ( 20 Data sets ) | 0 | 0.40 |
| Group H ( 20 Data sets ) | 0 | 0.45 |
| Group I ( 20 Data sets ) | 0 | 0.50 |
| Group L ( 20 Data sets ) | 0 | 0.60 |
| Group M ( 20 Data sets ) | 0 | 0.70 |
| Group N ( 20 Data sets ) | 0 | 0.80 |
| Group O ( 20 Data sets ) | 0 | 0.90 |
| Group P ( 20 Data sets ) | 0 | 1 |
| Group Q ( 20 Data sets ) | 0 | 1.20 |
| Group R ( 20 Data sets ) | 0 | 1.40 |
| Group S ( 20 Data sets ) | 0 | 1.60 |

**Tabella 3.1** Per testare il metodo basato su bootstrap sono stati generati 17 gruppi caratterizzati da diversi livelli di rumorosità e composti ognuno da 20 data sets. In totale sono dunque stati creati 340 data set.

## 3.2 Dati reali statici

Come descritto nel lavoro [22] sono stati esaminati 112 pazienti affetti da leucemia linfatica cronica (CLL), la forma leucemica più diffusa, diagnosticata presso la Divisione di Ematologia dell'Ospedale Niguarda di Milano. La diagnosi della patologia è stata basata su criteri morfologici standard e immunofenotipici. Al momento della diagnosi sono stati raccolti campioni utili per successivi studi su *marker* biologici come lo stato mutazionale dei geni che codificano per le catene pesanti della regione variabile delle immunoglobuline (*IgVh*) e il livello di espressione della proteina ZAP-70[1][23](tramite citometria).

I pazienti sono stati suddivisi in tre classi: la prima (n = 61) con geni con mutati e ZAP-70 negativo, la seconda (n = 28) caratterizzata da geni che codificano per l'IgVh non mutati e ZAP-70 positivo e la terza (n = 23) e la terza relativa a pazienti con ZAP-70 negativo e geni non mutati oppure ZAP-70 positivo e geni mutati. Da tutti i 112 soggetti sono state isolate cellule mononucleate del sangue periferico (*PBMCs*) e quindi purificati i linfociti-B secondo il protocollo esposto in [22] e conservate in una banca cellulare fino all'estrazione dell'RNA. Dall'RNA ottenuto si sono estratti i profili di espressione dei geni in accordo con il protocollo Affimetrix.

I valori dei profili di espressione per ogni *probe set* sono stati calcolati utilizzando un algoritmo *Robust Mullti-array Average* (RMA). Con l'algoritmo MAS 5-0 di affimetrix sono state scartate le probe set con valori marginali o assenti. Infine con l'algoritmo (SAM) *Significant Analysis of Microarrays* sono stati identificati i geni con variazioni di espressione statisticamente significative tra le diverse classi. Si sono così ottenuti 97 *probeset* significative corrispondenti a 65 geni selezionati usando un valore di soglia della *false discovery rate* (FDR) pari al 5%.

[1] ZAP-70 (Zeta-chain-associated protein kinase 70). Fa parte della famiglia delle proteine tirosin-chinasiche. Nei linfociti B è usata come marker pronostico per l'identificazione delle varie forme di CLL.

## 3.3 Dati reali dinamici[24]

Sono stati coltivati mioblasti L6 del muscolo scheletrico in un mezzo composto da una soluzione di glucosio (DMEM)[II] e siero fetale di bovino unita a una mistura di antibiotici e antimicotici. Sono state allestite quattro piastre per ogni campione temporale con una densità di 6:8 × $10^5$cells per 100 mm, tenute successivamente in incubazione nel mezzo di coltura per una notte. Il giorno seguente è stato cambiato il mezzo di coltura da DMEM per crescita cellulare a DMEM per differenziazione cellulare con l'1% di siero fetale di bovino. Il sesto giorno il processo di differenziazione era completato. La coltura è stata divisa in due gruppi: trattati con insulina e un gruppo di controllo. Le cellule sono state raccolte al tempo determinato nella fase di progettazione dell'esperimento facendo decantare la coltura, prelevandone la parte superiore e congelando immediatamente le cellule così estratte in azoto liquido. I campioni sono stati raccolti ogni 20 minuti per 8 ore sia per il gruppo trattato con insulina che il gruppo di controllo, per un totale di 50 campioni biologici. I campioni (20-40-60),(80-100-120),(140-160-180),(200-220-240),(260-280-300),(320-340-360),(380-400-420),(440-460-480) sia per il gruppo trattato che per il gruppo di controllo sono stati messi assieme ottenendo 8 campioni risultanti dall'unione. Questo per raggiungere un compromesso tra il costo dell'esperimento e la frequenza della griglia di campionamento: è più sicuro effettuare la media del segnale raggruppando i campioni biologici piuttosto che usare una griglia di campionamento sparsa (i geni a regolazione rapida verrebbero persi). L'intera procedura di campionamento e di coltura è stata ripetuta altre due volte in giorni differenti, utilizzando le medesime linee cellulari. I trascritti genici relativi ai due gruppi sono stati studiati tramite microarray di oligonucleotidi ad alta densità *Affimetrix* contenenti probe per 31099 geni. I passi di *preprocessing* [25] come la sottrazione del background, la normalizzazione delle *probe cell* e il calcolo dei livelli di espressione sono state effettuate usando la normalizzazione *quantile* e il software RMA. I geni sono stati pre-filtrati usando *Affimetrix Detection Cell* come visto per i dati statici (sezione 2.2). Le rimanenti 21958 *probeset* sono state

[II] Dulbecco's Modified Eagle Medium (DMEM) è un mezzo di coltura basale largamente utilizzato per supportare la crescita di molti tipologie di cellule di mammifero. http://products.invitrogen.com/.

ordinate sulla base dei espressione differenziale usando il metodo descritto in [26]. Le prime 1000 probe set verranno valutate nelle analisi.

# Capitolo 4

# Metodo

## 4.1 Bootstrap

Il metodo di riordinamento messo a punto nello studio [1] fornisce ordinamenti spesso molto buoni su dati affetti da basso livello di rumore mentre le sue prestazioni tendono inevitabilmente a peggiorare con il crescere dei disturbi. Ci si è dunque posti il problema di ricercare un metodo per migliorare la precisione degli ordinamenti ottenuti.

Il bootstrap è una tecnica metaeuristica, cioè una metodologia computazionale di ottimizzazione di una funzione costo che iterativamente aggiorna una soluzione proposta, basata sul ricampionamento dei dati al fine di migliorare le performance di classificazione o di regressione in termini di stabilità e accuratezza. Inoltre riduce la varianza e contribuisce a evitare l'overfitting. Nonostante il metodo sia usualmente applicato a modelli riguardanti alberi decisionali, può essere utilizzato per qualsiasi tipo di modello. Esempi di applicazioni algoritmiche del bootstrap sono il bagging e boosting, sui quali non ci soffermiamo in questa sede.

Dato un data set $D$ di dimensione N ( nel caso in esame N è i numero di profili, cioè geni, considerati ) il bootstrap consiste nella generazione di B nuovi data sets $D_i$, con $1 \leq i \leq B$, ognuno dei quali di dimensione $n'_i \leq N$,ottenuti ricampionando uniformemente con ripetizione gli N profili originari.

Il metodo di elaborazione dei dati (regressione o classificazione) scelto viene quindi applicato ai B data sets e i risultati che ne scaturiscono vengono in un secondo

momento combinati tra loro (ad esempio facendone la media). Un aspetto che limita l'utilizzo di questa tecnica è la complessità computazionale che cresce molto dal momento che il metodo deve essere reiterato per ognuno dei B data set (spesso B=100) e che i metodi per la combinazione successiva degli B risultati, o B insiemi di risultati, ottenuti possono presentare anch'essi una elevata complessità computazionale. È per questa ragione che si effettueranno prove con il bootstrap basate sul metodo deterministico e non probabilistico, eccessivamente oneroso.

Il bootstrap è stato implementato in matlab ricampionando ad ogni iterazione 1000 geni (dei 1000 geni che compongono ogni data set) con ripetizione e effettuando un numero di iterazioni pari a B=100.

In figura 4.1 è possibile veder il semplice codice della *function* utilizzata.

```
function[risultato_boot]=expression_bootstrap(matrice_expr,B,num_geni,num_soggetti)
%funzione che riceve in input:
%la matrice 1000x50 avente per righe i profili di espressione relativi ai 1000 geni
%il numero num_geni di geni da ricampionare con  ripetizione in ciascuna iterazione
%il numero B di iterazioni da effettuare-->determina
%l'onerosità computazionale e la stabilità del risultato
%il numero di campioni num_soggetti presenti per ciascun profilo
%di espressione (numero di colonne della matrice)
%ritorna in output:
%la matrice contente gli ordinamenti ricostruiti ad ogni iterazione
risultato_boot=zeros(num_geni,num_soggetti);
for i=1:B
    geni_considerati_temp=randint(1,num_geni,[1,1000]);
    matrice_expr_temp=matrice_expr(geni_considerati_temp,:);
    [ord_boot] = initOrder(matrice_expr_temp);
    risultato_boot(i,:)=ord_boot;
end
```

**Figura 4.1.**Codice MATLAB della funzione *expression_bootstrap* che effettua il bootstrap su dati di espressione genica (nel caso in esame simulati).

Una volta determinati tutti gli ordinamenti relativi ai vari passaggi iterativi il problema che si pone è di come integrarli tra loro. Per fare ciò si è scelto di ricorre alla teoria dei grafi impostando il problema come istanza del "traveller salesman problem" - TSP (problema del commesso viaggiatore) . Tale problema può essere così formulato:

→Dato un grafo completo pesato (cioè un insieme di V vertici e E archi, ed una funzione che restituisce il peso di ciascun arco sottoforma di numero reale: f: E ⟶ R, e

nel quale ogni vertice sia collegato a tutti i vertici rimanenti), trovare il cammino hamiltoniano (ossia il cammino che tocca tutti i vertici una e una sola volta) con peso minore, cioè per il quale sia minima la funzione costo:

$$J = \sum_{p \in P} f(p) \quad . \qquad \text{Eq. (4.1)}$$

In termini equivalenti, trovare il cammino hamiltoniano che minimizzi tale funzione costo:

$$P_{winner} = \text{argmin}_P [\, \mathbf{J} \,] = \text{argmin}_P \left[ \sum_{p \in P} f(p) \right] . \qquad \text{(Eq. 4.2)}$$

Mentre nella sua più tipica rappresentazione il problema viene formulato in questo modo: data una rete di città, connesse tramite delle strade, trovare il percorso di minore lunghezza che un commesso viaggiatore deve seguire per visitare tutte le città una e una sola volta. Nel caso in esame diventa: definito un grafo completo composto da N vertici o nodi (N è il numero di soggetti-campioni), E archi diretti rappresentativi di una relazione d'ordine tra i due soggetti-campioni che uniscono (vedi Figura 4.2) ai quali si associano altrettanti pesi così definiti:

W = pesi associati agli archi diretti E, rappresentativi del numero di occorrenze della relazione rappresentata dall'arco ai quali si riferiscono nei B ordinamenti ricostruiti nelle B iterazioni del bootstrap (possono assumere solo valori interi, e sono nulli nel caso non compaia in nessuno degli ordinamenti ricostruiti la relativa relazione d'ordine).

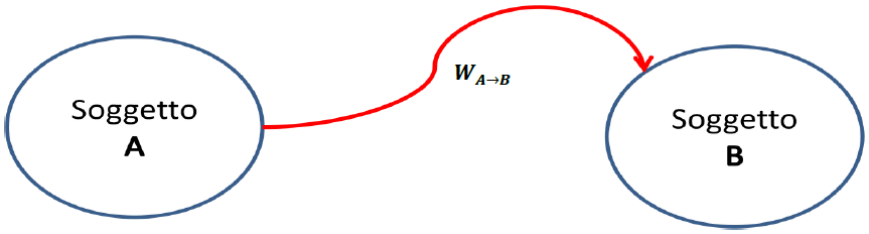



**Figura 4.2.** Elementi del grafo rappresentativi dell'istanza del problema TSP. Il peso assegnato all'arco A→B sarà tanto più elevato tante più volte comparirà nei vari ordinamenti la relazione d'ordine "A è immediatamente precedente a B". Se tale relazione non si verifica in nessuno dei B ordinamenti allora $W_{A \to B}$ assumerà valore nullo.

Per quanto riguarda la complessità computazionale va ricordato che TSP è un problema NP-difficile ("nondetermistic polynomial-time hard"), cioè che non esistono algoritmi efficienti per la sua risoluzione. L'unico metodo di risoluzione è rappresentato dall'enumerazione totale, ovvero nell'elaborazione di tutti i possibili cammini sul grafo per la successiva scelta di quello migliore. Tuttavia, la complessità dell'operazione la rende impraticabile per grafi di dimensioni comuni nei problemi reali: in un grafo di $n$ nodi, bisognerà calcolare, nel caso peggiore in cui ogni nodo è connesso con tutti gli altri, $n!$ ($n$ fattoriale) possibili cammini, il che implica una complessità esponenziale. È possibile fare ricorso a algoritmi euristici, cioè algoritmi che producono soluzioni probabilmente buone, in tempi computazionali più contenuti, ma che non è possibile provare siano ottimali.

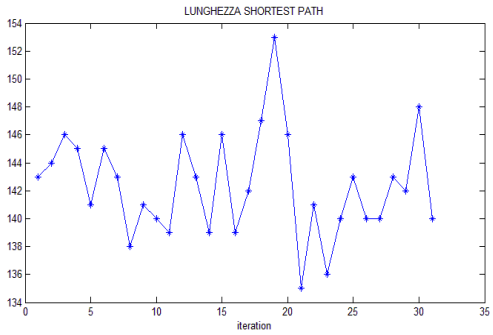



**Figura 4.3.** È possibile notare come varia la lunghezza degli *shortest path* praticando l'algoritmo più volte (nella fattispecie 30. In alcuni casi è evidente quale sia il cammino più breve, dato che partendo da diverse condizioni iniziali si ritrovo la stessa distanza (quindi verosimilmente il medesimo cammino). In altre situazioni, come quella in figura, la scelta del minore cammino minimo può essere meno lampante.

Entrando più nel dettaglio dell'implementazione è stata realizzata una funzione che realizza la matrice delle adiacenze ricevendo in input gli ordinamenti ottenuti con il bootstrap ottenendo come risultato parziale una matrice contenente il numero di occorrenze delle relazioni di ordinamento individuate dai rispettivi vertici. Dato che il problema deve essere posto in termini di ricerca di un cammino minimo la matrice deve essere modificata in modo che a valori bassi corrispondano un elevato numero di occorrenze e a valori elevati un basso numero di occorrenze. Nella figura 4.4 è disponibile una versione ridotta della funzione realizzata in MATLAB. Una volta realizzata tale matrice si applica l'algoritmo euristico e dato che il risultato dipende dalla configurazione iniziale delle connessioni scelta dall'algoritmo, è opportuno far girare il

codice più volte ( il numero scelto è 30 volte) scegliendo infine il cammino minimo corrispondente al percorso più breve (vedi fig.3.3).

```
%funzione che realizza una matrice delle adiacenze a partire dalla matrice
%ordinamenti realizzata con il bootstrap
function[matr_score_min_better,matr_score_max_better]=genera_matr_adiacenze(ordinamento_bootstrap)
%ordinamento_bootstrap=matrice(num_iterazioni_bootst X num_soggetti)
%ottenuta dal passo precedente di bootstrap
[B,num_soggetti]=size(ordinamento_bootstrap)
matrice_score=zeros(num_soggetti,num_soggetti);
for i=1:100
    ordinamento_corrente=ordinamento_bootstrap(i,:);
     for k=1:(num_soggetti-1)%aggiorno score osservando gli ordinamenti relativi in ordinamento_corrente
matrice_score(ordinamento_corrente(k),ordinamento_corrente(k+1))=(matrice_score(ordinamento_corrente(k),ordina
mento_corrente(k+1))+1);
    end
end

matr_score_max_better=matrice_score;
matr_score_min_better=ones(size(matrice_score))*max(max(matrice_score))-matrice_score;
```

**Figura 4.4.** Funzione R *genera_matr_adiacenze*.

Come implementazione MATLAB dell'euristica TSP si è scelto di usare un algoritmo genetico open source[27] che fornisce una soluzione del problema TSP open, cioè una variante del TSP per la versione non chiusa del problema: il commesso viaggiatore non torna alla città di partenza o nel nostro caso il primo elemento dell'ordinamento non coincide con l'ultimo. L'implementazione fornita riceve in input la matrice delle adiacenze e il numero di iterazioni eseguite dal programma e decise dall'utente, mentre ritorna in output l'ordine dei soggetti e la distanza minima associata al rispettivo cammino minimo, cioè il valore della funzione costo; inoltre genera i quattro grafici visibili in Figura 4.5. In particolare il grafico relativo alla "best solution history" guida nella scelta del numero di iterazioni che è utile far eseguire al codice.

In Figura 4.6, è possibile osservare la realizzazione di un grafo semplificato, ossia privato degli archi, cioè delle relazioni d'ordine meno frequenti e lo *shortest path* soluzione del TSP open.

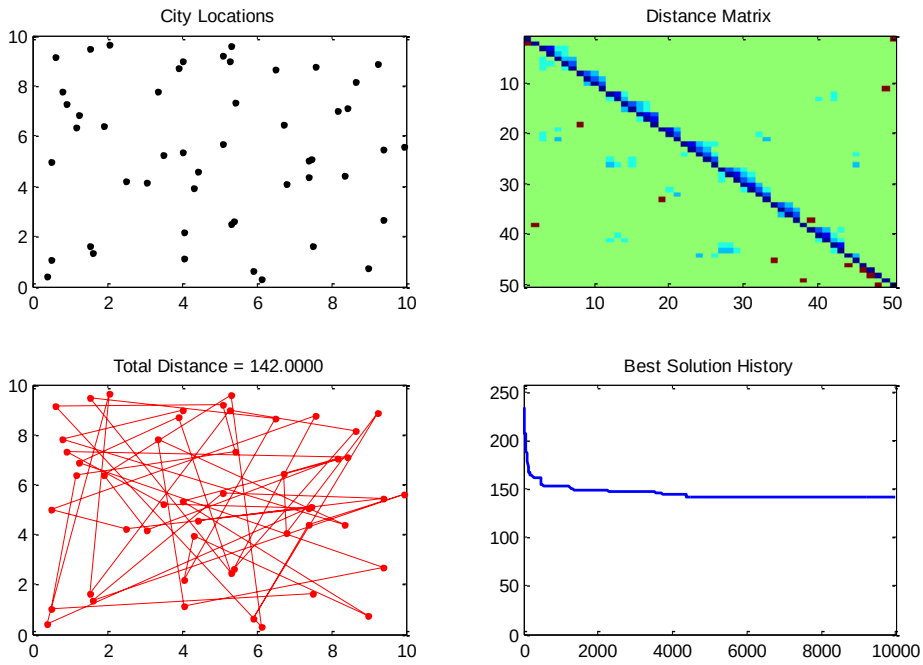



**Figura 4.5.** I quattro grafici generati dal codice che risolve il TSP open. Di particolare importanza in grafico in basso a destra che aiuta nella scelta del numero di iterazione ottimale e che quindi può rivelarsi utile nel ridurre il tempo computazionale, l'asse x infatti corrisponde al numero di interazioni, mentre l'asse y alla dinamica della soluzione.

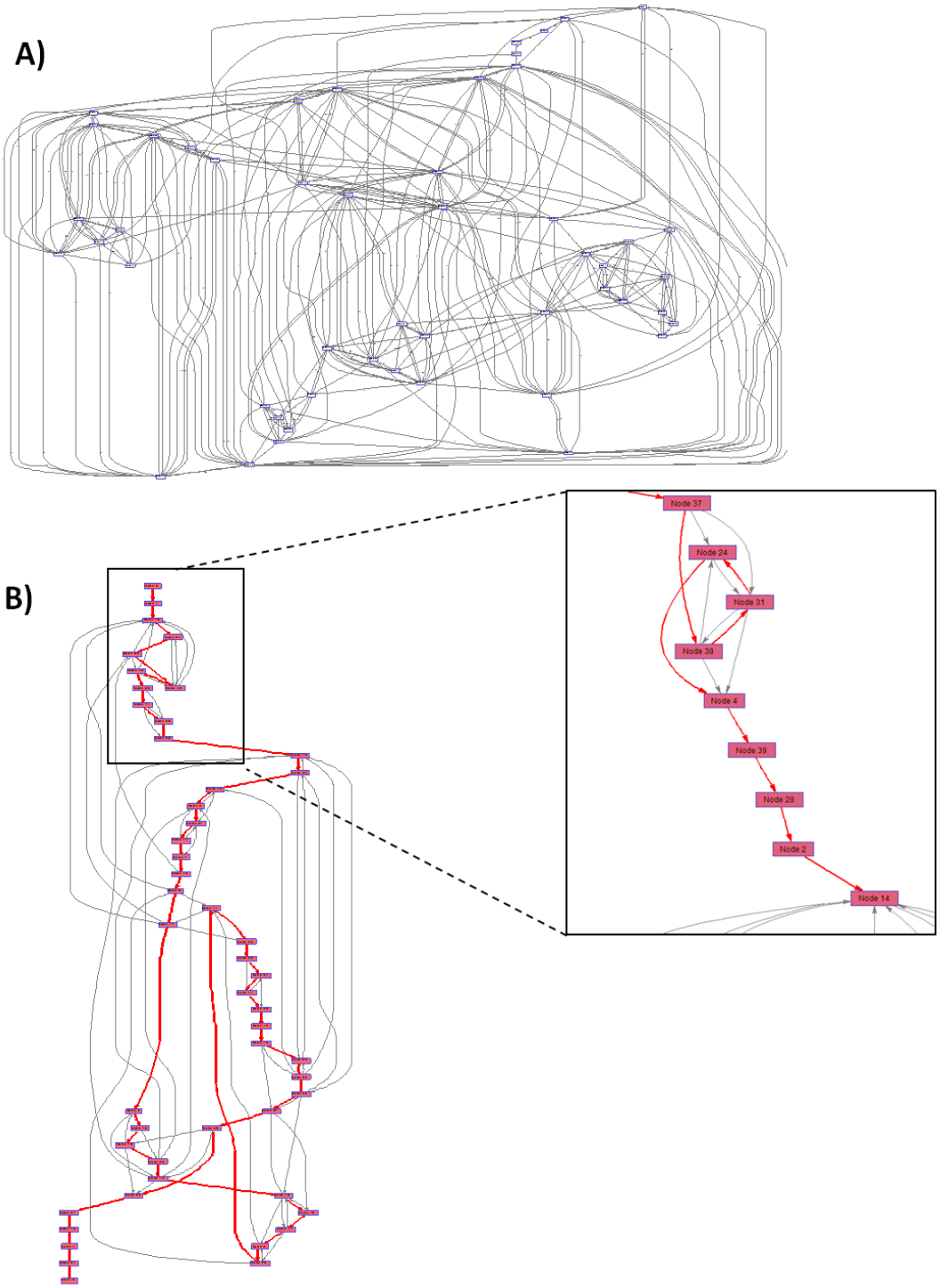



**Figura 4.6.** A) Mappa totale del grafo B) Grafo semplificato (privo degli archi meno significativi) del grafo con relativo shortest path in rosso.

## 4.2 Ordinamento in blocchi

Come già detto l'ordinamento ottenuto con il metodo visto nel Capitolo 1 spesso può non coincidere con l'ordinamento ottimale. I fattori che influiscono negativamente sono diversi. Senz'altro il rumore dal quale i dati sono affetti gioca un ruolo fondamentale e per considerare tale componente può essere utile l'adozione un modello probabilistico (vedi sezione 1.2 ). Inoltre il problema della ricerca dell'ordinamento temporale corretto anche nel verso, messo in questi termini, si rivela essere un problema mal posto (cioè le soluzioni ottenute sono in effetti più d'una). Il metodo visto fornisce infatti una relazione d'ordine non assoluta e solo l'introduzione di conoscenze a priori può permettere di discernere il verso temporale dei soggetti (ossia il verso di evoluzione temporale della malattia). Per chiarire questo concetto si pensi a soggetti affetti da una malattia degenerativa. Il metodo visto restituisce un ordine dei soggetti basandosi principalmente sulla minimizzazione delle distanze tra i valori di espressione assunti da campioni successivi. Dunque, utilizzando solo i dati di espressione, non è possibile stabilire se l'ordinamento ricavato sia da intendersi congruente all'evoluzione temporale della malattia (quindi i primi soggetti affetti dalla patologia ad uno stadio iniziale, mentre gli ultimi ad uno stadio avanzato) o se rispetto a questa siano in ordine inverso (cioè i primi soggetti e non gli ultimi corrispondenti ad uno stadio avanzato, mentre gli ultimi caratterizzati dalle prime fasi della malattia). Come è facile intuire, per innumerevoli ragioni, non ultime il dosaggio ottimale delle cure e la definizione di una prognosi accurata, la questione è di fondamentale importanza. Come introdotto poc'anzi solo l'introduzione di conoscenze a priori possono venire in aiuto e rendere il problema ben posto. Un esempio di informazione a priori che permetterebbe di scegliere la soluzione corretta delle due risultanti dal metodo, è la conoscenza, anche in termini generici, del comportamento del livello di espressione di almeno un gene lungo il decorso della malattia. Ad esempio, avere la certezza (derivata da esperimenti effettuati in precedenza) che anche solo uno degli M geni analizzati sia coinvolto nei processi derivanti dalla patologia, e che tale gene sia caratterizzato da un livello di espressione crescente (o decrescente) lungo il decorso della stessa, fornirebbe un parametro decisionale inequivocabile per la scelta del verso dell'ordinamento ricostruito. Un altro aspetto che va considerato e la cui evidenza risulta dalla trattazione di dati di espressione reali statici (Capitolo 5), è che la gran parte dei geni tipicamente considerati in

esperimenti statici non è coinvolta nella progressione temporale del fenomeno oggetto di studio (il decorso di una malattia), ed è dunque elemento di disturbo. Di conseguenza per poter ottenere una soluzione affidabile è fondamentale un'accurata fase di selezione dei geni. Per la trattazione di tale aspetto si rinvia il lettore al Capitolo 7.

L'ultimo problema da considerare, per il quale verrà proposta una trattazione nel seguito del capitolo, è che l'ordinamento fornito dal metodo spesso tende a ordinare a blocchi. Tali blocchi al loro interno potranno essere ordinati indistintamente in maniera diretta o inversa rispetto all'evoluzione temporale reale e in modo non per forza concorde tra loro. La risoluzione di tale problema con l'integrazione di conoscenza a priori, come fatto nello studio di riferimento [1] si rivela più ardua di quanto visto in precedenza per il problema del verso complessivo dell'ordinamento, dal momento che si dovrebbe conoscere il profilo del gene noto con una precisione decisamente più elevata. Inoltre, partire dal confronto con tale profilo noto sia per individuare i campioni-soggetti che delimitano i blocchi che per attribuire a questi in un secondo momento il verso corretto, oltre che essere un problema molto complesso soprattutto computazionalmente, si presta a ambiguità di interpretazione. Per queste ragioni si è proceduto proponendo un metodo di risoluzione del problema basato su tutt'altre ipotesi.

### 4.2.1 Identificazione dei blocchi

L'ipotesi di base è che i geni coinvolti nel processo oggetto di studio (quindi significativi) e con livelli di espressione caratterizzati da una peculiare evoluzione temporale (ad esempio crescente o decrescente), in corrispondenza ad un ordinamento in blocchi presentino delle discontinuità nel profilo di espressione e che tali discontinuità si riflettano nella derivata prima o seconda del profilo (vedi Figura 4.7).

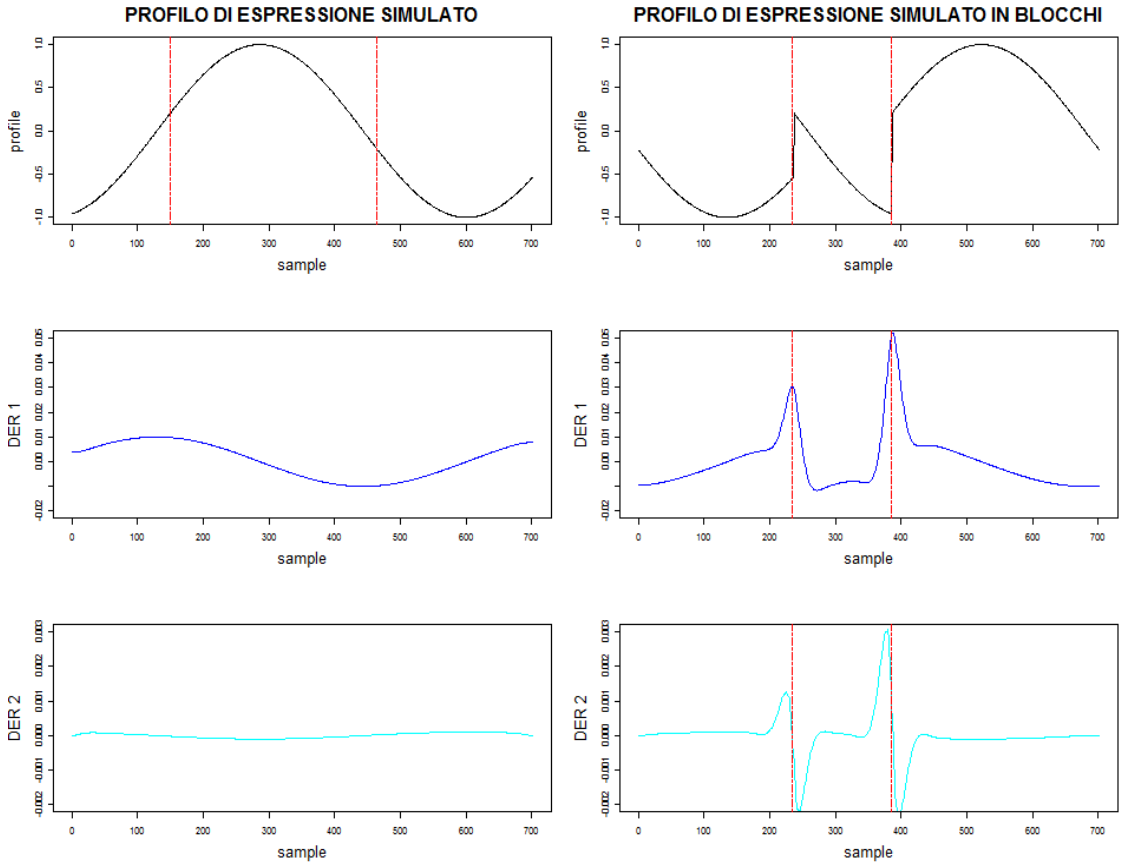



**Figura 4.7**.A sinistra: In alto: profilo di espressione del gene simulato; al centro: derivata prima del profilo di espressione; in basso: derivata seconda. A destra: come a sinistra ma relativo a profilo suddiviso in blocchi ricombinati tra loro.

Una volta rilevati i blocchi in seguito all'analisi della derivata è possibile effettuare tutte le combinazioni possibili tra tali blocchi e le rispettive versioni invertite e, tramite una seconda analisi della derivata, scegliere la combinazione alla quale corrisponde la derivata priva di picchi, o con picchi di minor ampiezza.

Nel caso di data set con un elevato numero di geni il discorso va esteso a tutti i rispettivi profili. Per poter considerare in una sola volta le derivate di tutti i geni si

possono scegliere varie strade, quella percorsa nel metodo proposto è probabilmente la più semplice. Considerando la matrice avente per righe i profili delle derivate dei vari geni, si è creato un unico vettore di lunghezza pari alla lunghezza dei profili, costruito facendo la somma per ogni colonna dei moduli di tali derivate, ottenute con l'ausilio delle SPLINE, che verranno viste più avanti in dettaglio. Si ottiene così una "derivata complessiva" rappresentativa delle continuità/discontinuità ricorrenti in tutti i profili. Per l'appunto è lecito attendersi che, nel caso della presenza di blocchi, le relative discontinuità si riscontrino nei campioni-soggetti vicini agli estremi dei blocchi, in molti dei profili significativi. Effettuando la somma per colonne si vanno a sommare tutte le discontinuità (Eq. (4.2)).

$$der\_compl_i = \sum_j \left| der_{ji} \right| \qquad \text{Eq. (4.2)}$$

Dove:

j è l'indice riferito alle righe della matrice contente le derivate, 1≤ j ≤N, N = numero geni;

i è l'indice riferito alle colonne della matrice e all'elemento del vettore "derivata complessiva", 1≤ i ≤M, M = numero soggetti-campioni.

Una soluzione senz'altro più elegante, ma che presenta qualche complessità in più è quella di pesare i valori dei moduli delle derivate di ogni singolo gene nell'effettuare la somma per colonne, attribuendo un peso legato al p_value ottenuto nella selezione dei geni differenzialmente espressi effettuata in precedenza, in modo da ridurre i disturbi. Si andrebbe infatti a pesare la derivata del profilo di espressione in base alla probabilità che il relativo gene sia o meno significativo. Così facendo ci si può attendere un miglioramento sul SNR.

Per la scelta del grado della derivata si rinvia il lettore all'Appendice del corrente Capitolo.

**Break joint e break point**

Per procedere nell'analisi dell'ordinamento in blocchi è utile soffermarsi a definire, con una formulazione che bene si presti ad un'implementazione algoritmica, il concetto

di blocco. Si parte facendo riferimento ad un solo gene, ossia al riordinamento dei campioni costituenti un solo profilo di espressione, sarà in seguito immediata l'estensione del concetto al caso di M profili di espressione (quindi M geni). In questo studio si definisce un blocco come un insieme di campioni-soggetti (nel seguito si farà riferimento indistintamente a campioni, soggetti e istanti temporali), appartenenti ad un ordinamento, consecutivi, caratterizzati da un'elevata regolarità dei profili nei valori da essi assunti, e delimitati da due campioni denominati break points, facenti parte del blocco stesso. Nel caso di ordinamento ottimo, l'ordinamento intero è l'unico blocco e il primo e l'ultimo campione sono gli unici break points che lo delimitano. Il numero di break points è chiaramente il doppio del numero di blocchi. Il primo e l'ultimo campione dell'ordinamento sono sempre break points. Si chiarisce ora il concetto di break joint. Due istanti temporali successivi compongono un break joint se i valori assunti dal profilo temporale nei due istanti sono molto diversi, cioè se la curva che approssima il profilo ha pendenza molto elevata (o molto bassa). Un break joint corrisponde all'approssimazione discreta del caso continuo in cui si ha un punto di discontinuità nella curva che descrive il profilo di espressione. Gli istanti temporali facenti parte di tutti i break joints di un ordinamento, più l'istante iniziale e finale, formano l'insieme dei break points dell'ordinamento. Ogni break joints è costituito dall'ultimo campione del blocco precedente e dal primo del blocco successivo. Per estendere ora i concetti di break joints e break points al caso di M geni-profili è utile riprendere alcuni concetti accennati in precedenza. Si consideri la matrice avente per righe le derivate discrete di ognuno degli M profili. Facendo una somma per colonne, come descritto in Eq. 4.2 si viene a creare un vettore di lunghezza pari al numero di soggetti, nel quale ogni elemento è costituito dalla somma dei valori assunti da ogni soggetto nei vari M profili di derivata, in modulo. È lecito supporre che in presenza di un break joints, in molti profili (pressoché tutti i profili relativi a geni significativi) la derivata assuma in modulo valori molto elevati che si andranno così a sommare tra loro.

Mentre per ciascun campione non delimitante un blocco, al più saranno presenti dei picchi in pochi profili dovuti alla normale dinamica di espressione, meno evidenti nel vettore della derivata complessivo. Una volta costruito tale vettore, in presenza di eventuali break joints ci si aspetta di osservare dei picchi, come visto in Figura 4.7. I

problemi da porsi sono: come rilevare tali picchi? Che ordine di derivata considerare? Una volta identificati i blocchi come ricomporre l'ordinamento corretto?

**Interpolazione SPLINE e i degrees of freedom**

Per l'implementazione del calcolo delle derivate si è ricorso alla funzione R *smooth.spline*, che realizza l'interpolazione *spline* cubica ( o di Hermite ) di un dato insieme di punti e restituisce la lista di punti ottenuti con l'interpolazione o la funzione che la esegue, e al metodo ad essa correlato *predict*. L'interpolazione *spline* è un particolare metodo di interpolazione basato sulle funzioni *spline*. Si tratta di uno strumento dell'analisi numerica utilizzato in molti campi applicativi. A differenza dell'interpolazione polinomiale, che utilizza un unico polinomio per approssimare la funzione su tutto l'intervallo di definizione, l'interpolazione *spline* è ottenuta suddividendo l'intervallo in più sotto-intervalli e scegliendo per ciascuno di essi un polinomio di grado d (solitamente piccolo) . Viene poi imposto che negli estremi di questi sotto-domini, detti nodi (*knots*), all'interno dei quali la funzione viene rappresentata da un polinomio di ordine d, vengano inoltre soddisfatte le condizioni di continuità della funzione e delle sue derivate di ordine $(d - 1)$, cioè che due polinomi successivi si saldino in modo liscio. La funzione risultante apparterrà quindi alla classe:

$$C^{(n-1)[x_1,x_D,]}.$$

La funzione che si ottiene con un procedimento di questo genere si chiama funzione *spline*. L'interpolazione lineare, che utilizza una funzione lineare, ossia un polinomio di grado 1, su ogni sotto-intervallo può essere considerata un caso particolare di interpolazione *spline*. La funzione interpolante ottenuta con l'interpolazione *spline* è più liscia di quelle ottenute con altri metodi ( ad esempio con l'interpolazione polinomiale ) , nel senso che è la funzione interpolante con curvatura media minima. Inoltre, l'interpolante *spline* risulta più facile da valutare dei polinomi di grado elevato richiesti dalla interpolazione polinomiale e non soffre del fenomeno di Runge, che consiste nell'aumento di ampiezza dell'errore in prossimità degli estremi dell'intervallo. Tuttavia, se i dati da interpolare hanno conformazioni particolari ( ad esempio formano dei gradini ) , la *spline* interpolante può essere soggetta al fenomeno di Gibbs, ampie oscillazioni in vicinanza di un gradino (Figura 4.8). Per ovviare a questo problema

vengono utilizzate le *smoothing spline*. Le *spline* possono inoltre sfruttare agevolmente formule di ricorrenza e, poiché sono costituite da tratti di polinomi, sono utilizzabili direttamente per calcolare derivate ed integrali. Poiché nella letteratura non sempre si opera una chiara distinzione tra nodi e punti sperimentali, è bene ricordare che, essendo la *spline* definite a tratti, è necessario individuare il relativo supporto; i punti che formano il supporto sono chiamati nodi e nel caso delle *spline* il supporto coincide con i punti sperimentali.

Un parametro fondamentale è il numero di gradi di liberta df (*degrees of freedom*) equivalente desiderato (traccia della matrice *smoother*), che va a determinare il grado di *smoothing* della regressione (vedi Figura 4.9). Per la scelta del valore più adatto da attribuire al parametro *degrees of freedom* nell'implementazione del metodo rinviamo il lettore all'Appendice dell'attuale Capitolo.

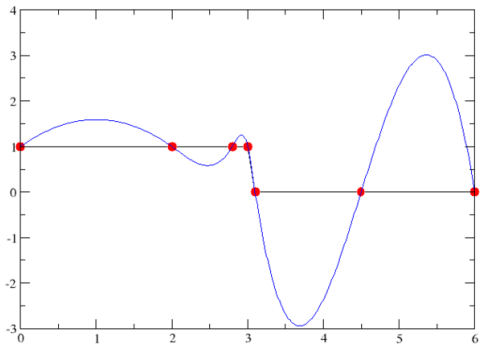


**Figura 4.8**. Fenomeno di Gibbs nell'interpolazione di un "gradino" tramite una *spline* cubica.

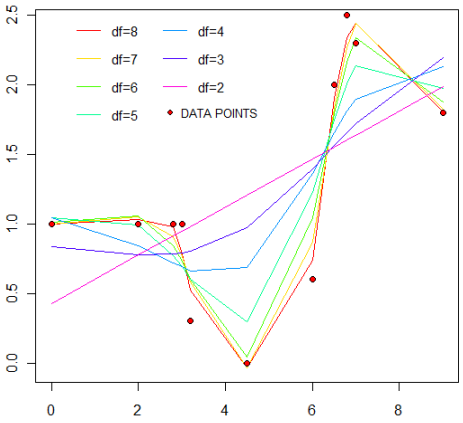



**Figura 4.9**. Relazione tra smoothness e degrees of freedom.

### Determinazione dei break joint

Una volta ricavato il profilo della derivata complessiva come spiegato in precedenza, la questione da risolvere è come determinare i campioni corrispondenti ai valori di picco, quindi identificare i *break joints* e dunque i blocchi. La soluzione adottata in questo studio è stata quella di effettuare un procedimento di clustering monodimensionale sui valori assunti dai campioni, per la precisione sui valori assunti dalla derivata discreta assoluta complessiva dei profili di espressione; detto in altri termini determinare un sottoinsieme dei capioni-soggetti caratterizzati da valori della derivata complessiva sensibilmente (e statisticamente) superiori alla media. Una volta effettuato il *clustering* va considerato l'insieme caratterizzato da valore del centroide più elevato. Una fatto da tener presente è che questo passaggio va eseguito una sola volta per data set da analizzare e dunque è consigliabile, nel caso in cui non sia abbia a che fare con un numero elevato di data set, una validazione diretta da parte dell'utente dei *break joints* individuati, dal momento che la procedura di *clustering* è una fase molto delicata del metodo proposto. A tale scopo è stata implementata una funzione che propone all'utente la scelta dei campioni ottenuti tramite la procedura di *clustering*, ma che attende la validazione dell'utente aiutato dalla visualizzazione della derivata, e da una serie di dati statistici come la media dei valori del cluster dei picchi scelto dal *clustering*, la media dei valori della derivata complessiva, la media dei valori relativamente alla media della derivata complessiva in percentuale (vedi figura 4.10). Nel caso di una validazione sperimentale del metodo, statisticamente valida, il numero di data set è chiaramente molto elevato e dunque è necessario utilizzare una procedura automatica per effettuare l'identificazione dei blocchi. Bisogna inoltre fare attenzione al fatto che il *clustering* non restituisce i campioni rispondenti alla definizione di *break points* o *break joint* e dunque è il caso di inserire dei controlli in modo tale da ottenere un insieme di campioni a due a due consecutivi, (consecutivi *solo* a due a due altrimenti si avrebbe la formazione di blocchi nulli), oltre al primo e ultimo elemento dell'ordinamento. La tipologia di *clustering* utilizzata per individuare i candidati break point è il *k-means* e i parametri sono stati settati dopo valutazioni sperimentali su dati simulati, in termini di *precision* e *recall*. I dettagli sono presenti in Appendice a questo Capitolo. Mentre una

trattazione elementare riguardante le metodologie di clustering, in particolare le due utilizzate in questo studio: k-means e gerarchico è fornita nella sezione 4.3.

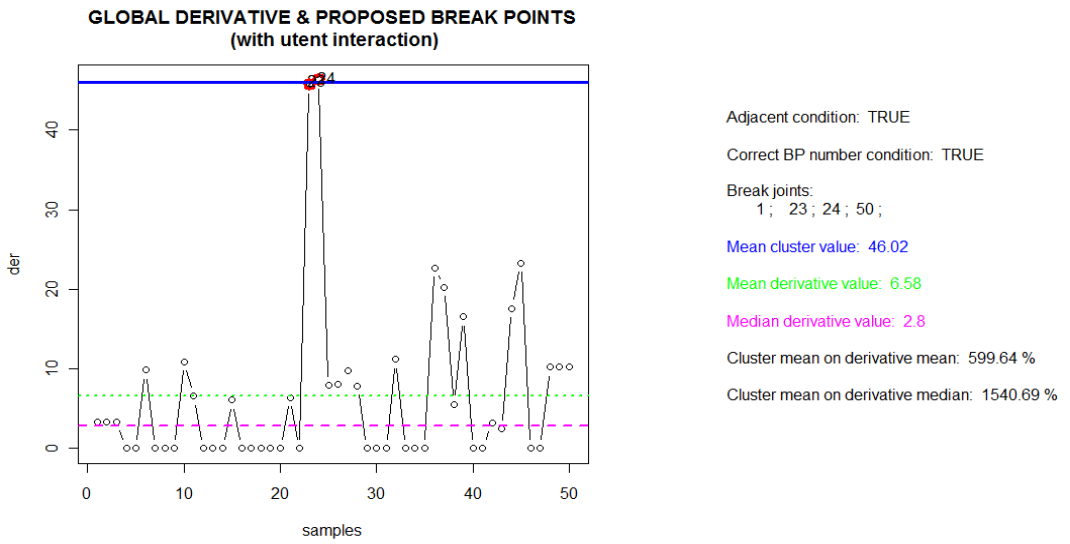



**Figura 4.10**. Esempio di grafico ottenuto utilizzando la versione interattiva del metodo di identificazione dei blocchi. Tramite il *clustering* monodimensionale vengono suggeriti dei *break points* (cerchiati in rosso). La linea blu corrisponde al valor medio (centroide) del cluster, la linea verde tratteggiata al valor medio del profilo della derivata globale, la linea rosa alla mediana dello stesso profilo. Sulla destra si trovano una serie di parametri utili a valutare la bontà della scelta del *cluster* dei *break points*. La condizioni di adiacenza e di numero corretto corrispondono alla verifica che i *break points* identificati siano consecutivi a due a due e solo a due a due e siano in numero pari. La procedura chiede all'utente di effettuare la scelta dei *break points* finché non sono soddisfatte le condizioni.

### 4.2.2 Ricombinazione dei blocchi

Una volta individuati i blocchi, come spiegato nella sezione precedente, bisogna: effettuare tutte le combinazioni possibili, eliminando le naturali ridondanze, ricomporre la matrice contente i dati di espressione riordinati (nel nostro caso con soggetti ordinati tramite il metodo deterministico esposto nel Capitolo 1) secondo la nuova combinazione dei blocchi e confrontare tra loro le derivate complessive assolute corrispondenti alle varie matrice di espressione relative a tutte le combinazioni ottenute dei blocchi; per effettuare tale confronto le possibili strade da intraprendere per la scelta di una sola delle combinazioni possibili sono varie. Nello studio qui esposto si è scelto di utilizzare ancora una volta il clustering effettuando una sorta di percorso a ritroso di quanto visto per l'identificazione dei blocchi. Se in precedenza si è scelto di identificare

i blocchi come delimitati dai campioni corrispondenti al cluster con centroide più elevato, ora si sceglie la combinazione per la quale il clustering effettuato sui valori della derivata complessiva ritorna un cluster "massimo" con centroide più basso.

Per chiarire meglio i passi seguiti si descrivono nel seguito i passi seguiti nell'implementazione del "*blok reordering*":

1) Dato un numero N di blocchi individuati si generano tutte le possibili combinazioni di blocchi e di verso.

Il numero di possibili combinazioni di blocchi è N!, mentre le possibili scelte di direzione, che per ogni blocchi possono essere due (diretta e inversa) sono in numero pari a $2^N$. Dunque le possibili combinazioni complessive sono:

$$numero\_possibili\_combinazioni = N! * 2^N.$$

Osservando la Figura 4.11 si può notare come il costo computazionale tenda ad esplodere anche per un numero di blocchi basso; pur considerando la presenza di combinazioni ridondanti (vedi punto seguente) si riesce a dimezzare il numero di combinazioni il costo computazionale derivante è comunque eccessivo per un numero di blocchi superiore a 5/6.

2) Una volta ottenute le possibili combinazioni di blocchi e direzione si traduce questa informazione in termini di *break points*. Cioè si genera una matrice avente numero di colonne pari al numero di *break points* (cioè pari al doppio del numero di blocchi) e numero di righe pari al numero di combinazioni totali. Ogni riga della matrice deve contenere i campioni break points ordinati secondo la rispettiva combinazione di blocchi e direzione.

Ad esempio dato il vettore di break points:

$\text{vett\_BP} =< 1,12,13,20 >$,

la combinazione sui blocchi:

$\text{comb\_blocchi} =< 2,1 >$,

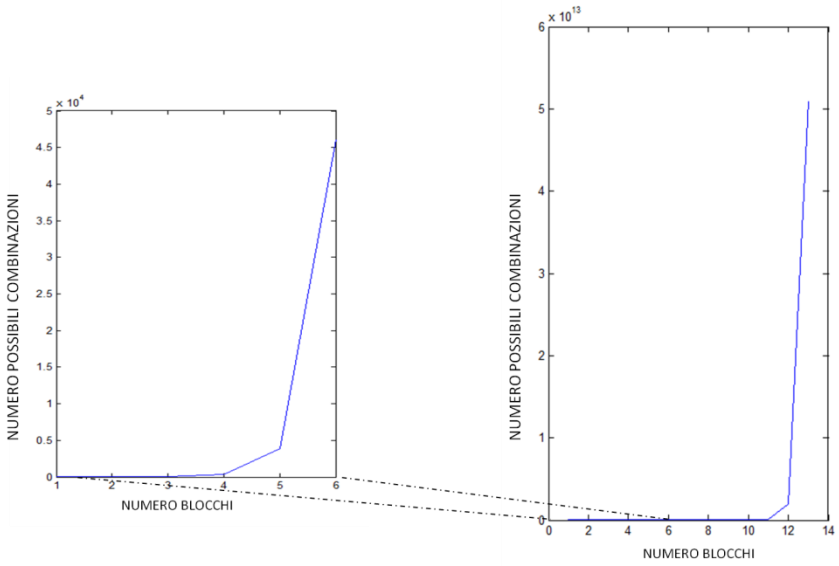
e su direzione:

$\text{comb\_direz} =< "D","I" >$,

la corrispondente riga della matrice dei *break points* ricombinati dovrà essere:

$mat\_BP\_ricomb_i =< 20,13,1,12 >$, con i= indice riga della matrice.

**Figura 4.11.** Numero di combinazioni possibili di blocchi e direzioni. Si nota come il numero di combinazioni tenda ad esplodere rendendo il problema in pratica non trattabile per un numero blocchi superiore al più alla decina (in realtà già con 5 blocchi il problema si rivela estremamente oneroso).




3) Una volta definita la matrice dei *break points* riordinati eliminare le ridondanze risulta molto semplice, basta eliminare le righe che si ripetono uguali (in R è possibile utilizzare l'utile comando *unique*). Questa operazione permette, come anticipato in precedenza, di dimezzare il numero di possibili combinazioni, cioè il numero di righe della matrice dei *break points* riordinati.
4) Per ogni riga della matrice dei *break points* così ottenuta, cioè per ogni combinazione di blocchi e direzione, bisogna ricavare una matrice contenete i profili di espressione dei geni (sulle righe) relativi a campioni riordinati secondo la rispettava combinazione. La matrice ottenuta avrà dimensioni pari alle dimensioni della matrice di espressione iniziale.

   Nell'esempio visto sopra l'ordine delle colonne dovrà essere:

   $$ord\_col =< 20,19,..,14,13,1,2,..,11,12 >,$$

   si ottengono dunque tante matrici quante sono le combinazioni.
5) Data un numero M di matrice di espressione relative a soggetti riordinati, con M numero di combinazioni possibili prive di ridondanze, si ricava la rispettiva derivata globale assoluta, come descritto nella sezione precedente.

6) Effettuiamo il *clustering* monodimensionale sui valori assunti dalla derivata, isolando il *cluster* corrispondente al centroide con valore maggiore. Per i dettagli sulla tipologia di *clustering* scelta e sui valori dei parametri si rinvia il lettore all'Appendice al Capitolo. Si tiene memoria di tale valore e della corrispondente combinazione.
7) Si confrontano gli M valori dei centroidi dei cluster più elevati (presumibilmente i cluster contenenti i valori di picco) e si sceglie la combinazione di blocchi e direzioni corrispondente al cluster più elevato, tra i cluster in cui è sono stati suddivisi i campioni, avente valore del centroide minore. L'idea è che si scelga la combinazione *winner* come quella combinazione avente meno picchi o meglio picchi di entità più ridotta. Cioè quella combinazione alla quale corrisponde un profilo della derivata più regolare, a sua volta associabile a profili privi di forti discontinuità, sintomi della presenza di blocchi. Nel caso in cui non fossero presenti blocchi si presume che il metodo ritrovi come combinazione *winner* la combinazione di partenza, nel caso di numero blocchi =2 visto sopra:

   $\text{winner\_comb\_blocchi} =< 1,2 >,$

   $\text{winner\_comb\_direzione} =< D, D >,$

   o indifferentemente

   $\text{winner\_comb\_blocchi} =< 2,1 >,$

   $\text{winner\_comb\_direzione} =< I, I >.$

8) Si riorganizza la matrice di espressione iniziale in modo tale da rispecchiare l'ordinamento *winner*

## 4.3 Clustering

Il *clustering* è un metodo, o meglio un insieme di metodologie, di "*Data Mining*" che, nel caso di applicazioni in ambito genomico, può essere definito come il processo di suddivisione di un insieme di geni in diversi sottoinsiemi (cluster) sulla base di relazioni di similarità tra pattern.

Il termine "*Data Mining*" è basato sull'analogia delle operazioni dei minatori che "scavano" all'interno elle miniere grandi quantità di materiale di poco valore per trovare l'oro. Nel DM questo "oro" è l'informazione, precedentemente sconosciuta e non distinguibile, il materiale di poco valore sono i dati e le operazioni di scavo sono le tecniche di esplorazione dei dati.

Il DM si può collegare a vari settori del sapere come la Teoria dell'Informazione, il Calcolo Numerico, l'Intelligenza Artificiale (in particolare *machine learning, pattern recognition*), la Statistica Metodologica (con particolare riferimento alla statistica computazionale e multivariata), il Calcolo delle Probabilità, e le discipline economico-aziendali, specialmente nell'ambito del marketing dell'organizzazione aziendale.

L'obiettivo del *clustering* su dati di espressione genica provenienti da esperimenti su *microarray* è quello di definire *clusters* in modo da minimizzare la variabilità intracluster e massimizzare la distanza intercluster; in altri termini trovare gruppi di geni composti da membri simili tra loro, ma distanti dai membri degli altri *cluster*, sulla base dei *pattern* di espressione dei geni, utilizzando una misura di similarità o distanza decisa a priori. Ci sono metodologie che invece di basarsi su misure di similarità o distanza fanno ricorso a modelli probabilistici, ad esempio il *clustering* a massima verosimiglianza (maximum likelihood) o il *clustering* Bayesiano.

È possibile ricorrere a due strategie di *clustering*: supervisionata (*supervised*), basata su conoscenze a priori, e non supervisionata (*unsupervised*). Gran parte dei progetti di DM sono supervisionati e il loro obiettivo è quello di generare previsioni, stime, classificazioni o caratterizzazioni relativamente al comportamento di alcune variabili target già individuate in funzione di variabili di input. Ovvero, nei metodi di apprendimento supervisionato, il data set contiene l'etichetta che indica la classe da apprendere e i nuovi dati sono classificati sulla base di quello che l'algoritmo apprende. In questa sede non ci si occuperà delle metodologie di *clustering* supervisionato, dal

momento che per i dati sui quali si vuole adoperare il metodo di *reordering* non sono disponibili etichette per i geni e dunque si rivela impossibile da effettuare la fase di apprendimento. Le tecniche di *clustering* che verranno brevemente introdotte nel seguito della sezione sono le due strategie algoritmiche di *clustering* non supervisionato utilizzate più frequentemente e prendono il nome di *clustering* gerarchico e *clustering k-means*.

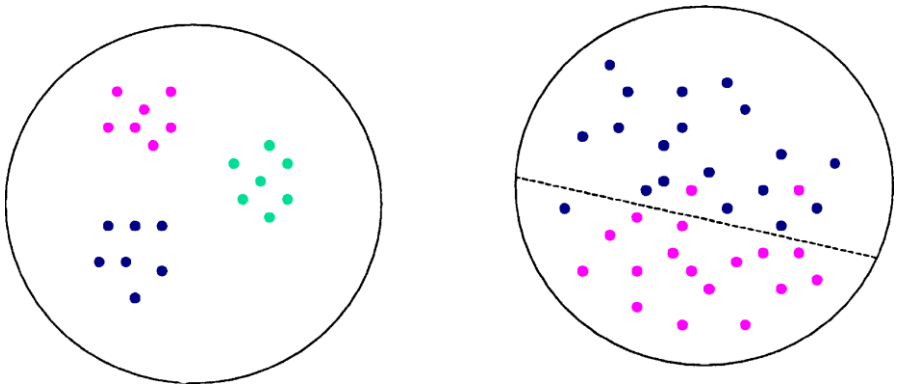


**Figura 4.12**. Analisi dei dati supervisionata e non supervisionata. A sinistra: nel caso non supervisionato vengono forniti dei *data points* in uno spazio n dimensionale ( in questo caso n=2) e l'obiettivo è quello di raggruppare assieme punti aventi caratteristiche simili. In questo caso ci sono tre *clusters* naturali, ognuno costituito da *data points* vicini tra loro nel senso di distanza euclidea. Un algoritmo di *clustering* dovrebbe riuscire a identificare tali tre *clusters*. A destra: nel caso supervisionato, gli elementi sono etichettati e l'obiettivo è essere in grado di determinare un insieme di regole di classificazione che permettano di determinare il gruppo di appartenenza di un nuovo punto fornito con la massima precisione possibile. Nell'esempio linea tratteggiata rappresenta una separazione lineare tra classi diverse.

Il *clustering* gerarchico è una procedura non supervisionata che data una matrice di distanza, risultato di confronti tra coppie di elementi, la trasforma in una struttura gerarchica, che può essere rappresentata per mezzo di un dendrogramma ad albero. che parte dalle foglie, nel caso in questione i geni, e va fino ad un super-cluster che contiene tutti i geni (vedi Figura 4.13). Gli algoritmi di *clustering* gerarchico sono di due tipologie: metodi agglomerativi e metodi divisivi. Il primo è di tipo *bottom-up*, cioè per fusioni successive delle entità elementari (n cluster foglia ognuno costituito da un gene) in cluster via via più grandi fino ad ottenere un solo cluster contente tutti i geni. Il secondo

inizia con un solo cluster e iterativamente definisce un nuovo cluster, in modo da ridurre il più possibile l'eterogeneità degli elementi.

Quando è possibile trovare una ragionevole definizione di distanza tra cluster diversi, la procedura agglomerativa presenta un costo computazionale minore rispetto alla procedura divisiva. Ha però lo svantaggio che un incorretta fusione di *cluster* nei passi iniziali spesso conduce a risultati molto distanti dalla reale suddivisione. La procedura divisiva inizia invece agglomerando e arrangiando i cluster più interessanti e dunque si rivela molto più robusta. Solitamente viene preferita la procedura agglomerativa per la sua efficienza.

Quanto detto in questa sezione sul *clustering* si può estendere anche al caso monodimensionale visto nella sezione precedente, in cui l'obiettivo non è quello di raggruppare profili di espressione simili, ma quello di identificare i campioni corrispondenti ai valori di picco di un profilo (costituito dalla derivata globale), permettendo la successiva identificazione dei blocchi presenti nell'ordinamento ricavato.

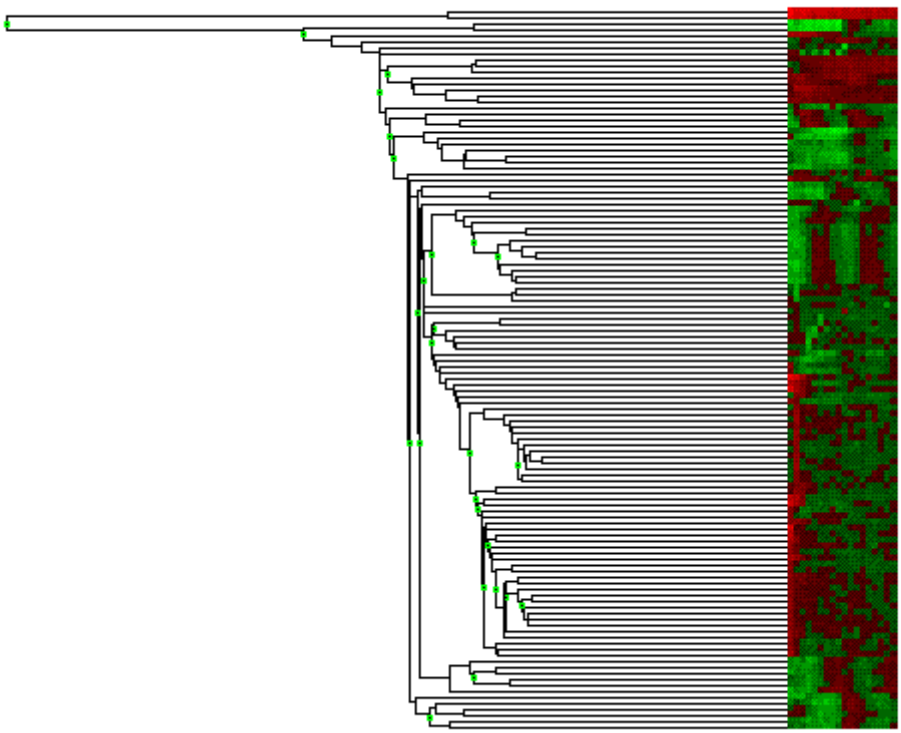


**Figura 4.13.** Esempio di struttura gerarchica (dendrogramma) utilizzata per descrivere il risultato di una procedura di *clustering* gerarchico su geni.

## 4.4 Valutazione delle performance

Nel caso di dati simulati, di dati ottenuti da serie temporali o più genericamente di campioni il cui ordinamento temporale reale è noto, una volta applicato un metodo di riordinamento, si necessita la definizione di un valore da attribuire all'ordinamento ottenuto per valutarne la bontà. Dal momento che ogni metodo qui proposto non è in grado di ricostruire un ordinamento in termini assoluti, ma solo di stabilirne uno relativo tra i campioni-soggetti, non sarà sufficiente confrontare la sequenza temporale ottenuta con quella reale, cioè ottenere un punteggio riferito alla sequenza originale in ordine diretto. Bisognerà invece valutare anche la distanza tra l'ordinamento ricostruito e la sequnza di campioni relativa all'ordinamento "vero" in ordine inverso. Inoltre, dal momento che non ha senso pretendere che il metodo applicato sia in grado di ricostruire l'ordinamento nella direzione corretta, ossia discernere la direzione dell'evoluzione temporale, il punteggio che ha senso osservare per stabilire la bontà del procedimento sarà il minore dei due appena citati.

Si procede ora definindo un primo punteggio che tiene conto solamente della distanza tra la posizione nell'ordinamento ricostruito del campione i-esimo, con $1 \leq i \leq N$ (dove N è il numero di campioni-soggetti), e la posizione del medesimo campione i-esimo nell'ordinamento "vero".

$$DIST_i = |true\_position_i - recovered\_position_i| \; ; \quad \text{Eq. (4.3)}$$

Ad esempio sia:

$vect_{TRUE}$=<5,2,3,1>, la sequenza temporale dei campioni reale;

e sia:

$vect_{RECOVERED}$=<2,3,5,1>, la sequenza ricostruita dal metodo di riordinamento.

Le distanze che si ottengono, che ovviamente possono assumere solo valori interi, sono:

$$DIST_1 = \left|position\left(vect_{TRUE_1}\right) - position\left(vect_{RECOVERED} = vect_{TRUE_1}\right)\right| =$$
$$|1 - 3| = 2$$

$$DIST_2 = \left|position\left(vect_{TRUE_2}\right) - position\left(vect_{RECOVERED} = vect_{TRUE_2}\right)\right| = |2 - 1| = 1$$

$$DIST_3 = \left|position\left(vect_{TRUE_3}\right) - position\left(vect_{RECOVERED} = vect_{TRUE_3}\right)\right| = |3 - 2| = 1$$

$$DIST_4 = \left|position\left(vect_{TRUE_4}\right) - position\left(vect_{RECOVERED} = vect_{TRUE_4}\right)\right| = |4 - 4| = 0$$

Il significato dei valori assegnati è facilmente intuibile: ad un punteggio elevato corrisponderà una notevole differenza tra le posizioni nei due ordinamenti dello stesso campione, mentre si ottiene un punteggio basso nel caso in cui la posizione nei due ordinamenti sia simile ( 0 se la posizione nei due ordinamenti è la medesima).

Dalla definizione di un punteggio associato ad ogni singolo campione ad un punteggio associato all'intero ordinamento ricostruito ( sempre in riferimento ad un ordinamento "vero"), il passo è breve. È immediato definire l'errore:

$$\mathbf{ER}(vect_{RECOVERED} \mid vect_{TRUE}) = \sum_{i=1}^{N} DIST_i \ ; \qquad \text{Eq. (4.4)}$$

dove $1 \leq i \leq N$ (con N numero di campioni-soggetti cioè la lunghezza dei vettori ordinamento).

Nell'esempio visto in precedenza si ottiene:

$$\mathbf{ER}(vect_{RECOVERED} \mid vect_{TRUE}) = \sum_{i=1}^{N} DIST_i = 2 + 1 + 1 + 0 = 4;$$

L'errore così definito è evidentemente sensibile al numero di elementi dell'ordinamento, cioè dal numero di soggetti-campioni, e ha dunque bisogno di un'ultima modifica per poter essere utilizzato come parametro di confronto tra ordinamenti ricostruiti relativi a data set di diverse dimensioni. Per rendere ordinamenti di diversa lunghezza (con un numero di soggetti-campione diverso) confrontabili si è proceduto dividendo l'errore definito in Eq. 4.4 per il massimo errore raggiungibile ("*worst case*") da ordinamenti aventi un determinato numero di campioni. Per definire il

"worst case" basta considerare che l'errore ER più elevato, dato il numero di elementi N dell'ordinamento, si ha nel caso di ordinamento ricostruito specularmente invertito rispetto a quello vero. Per esempio considerando i 4 campioni dell'ordinamento rappresentato dal vettore $vect_{TRUE}$ , il peggior ordinamento ottenibile è rappresentato dal vettore:

$$vect_{WorstCase} =< 1,3,2,5 >$$

Per il quale si otterrà:

$$\mathbf{ER}(vect_{WorstCase} \mid vect_{TRUE}) = \sum_{i=1}^{N} DIST_i = 3 + 1 + 1 + 3 = 8 \ ;$$

empiricamente si nota che il "*worst case score*" ($\mathbf{ER}(vect_{WorstCase} \mid vect_{TRUE})$ , che per semplificare in seguito verrà indicato con la notazione $\mathbf{ER}_{WC}$, segue la regola:

$$\mathbf{ER}_{WC} = \big((N-1) + (N-3) + \cdots\big) * 2) \quad ;$$

che è stata implementata in R con il codice di Figura 4.14.

```
max_distance<-function(ordine_riferimento,ordine_da_confrontare)
{#questa function ritorna la massima distanza possibile tra due
 #ordinamenti di eguale lunghezza utilizzando questa function
 #è possibile relativizzare la quantificazione della distanza tra
 #due ordinamenti rispetto alla loro lunghezza ossia al
 #numero di campioni da ordinare
 #questa funzione viene chiamata dalla funzione eval_order
 n<-length(ordine_riferimento)
 aus_num_term=n%/%2
 somma<-0
 l<-(n-1)
 for (i in (1:aus_num_term))
  {
    somma<-(somma+l)
    l<-(l-2)
  }
  max_dist<-(somma*2)
return(max_dist)
}
```

**Figura 4-14.** Implementazione in R della funzione che calcola il valore dello errore riferito all'ordinamento "*worst case*".

L'errore che verrà utilizzato per valutare la bontà di un ordinamento si definisce con la seguente formulazione conclusiva:

$$\boldsymbol{ER_{final}}(vect_{RECOVERED} | \, vect_{TRUE}) = \frac{\mathbf{ER}(vect_{RECOVERED} | \, vect_{TRUE})}{\mathbf{ER}_{WC}} = \frac{\sum_{i=1}^{N} DIST_i}{\mathbf{ER}_{WC}} \quad ;$$

Eq. (4.5)

Va notato che la formulazione proposta corrisponde a relativizzare l'errore rispetto al numero di soggetti-campioni, dunque si può parlare di errore relativo.

Un errore di valore 0 corrisponde ad un ordinamento perfetto mentre un errore uguale a 1 ad un ordinamento speculare a quello di riferimento. Ritornando all'esempio considerato in precedenza l'errore, come definito in Eq. 4.5, che si ottiene è:

$$\boldsymbol{ER_{final}}(vect_{RECOVERED} | \, vect_{TRUE}) = 4/8 = 0.5 \ .$$

In fig. 4.15 si possono vedere degli esempi di profili di espressione ricostruiti, ottenuti da dati simulati, corrispondenti a vari errori relativi.

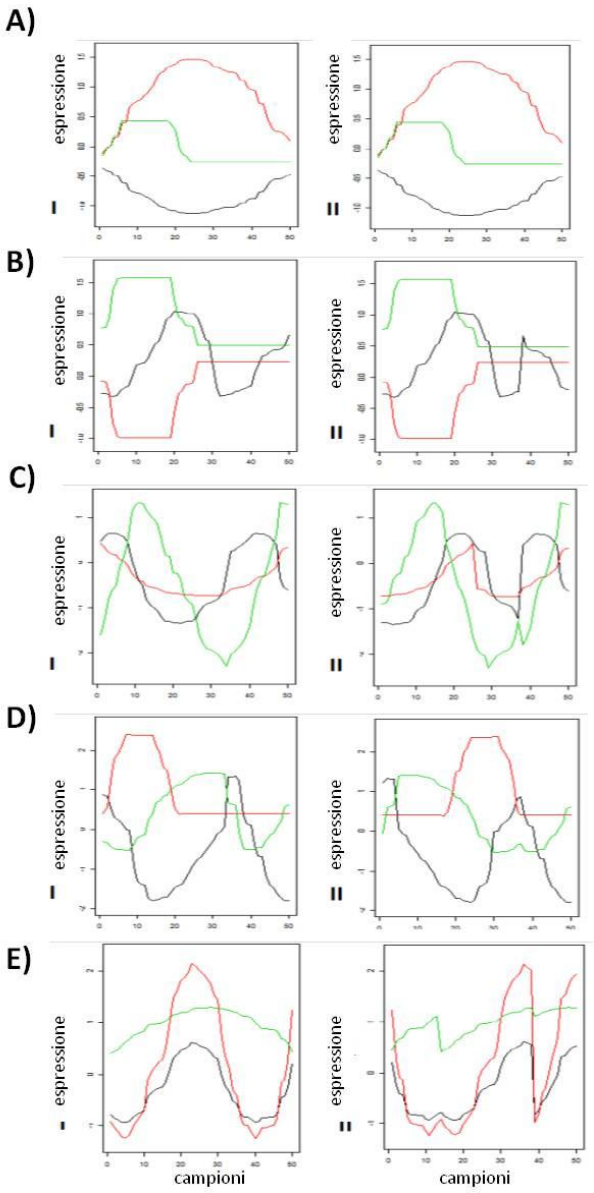



**Figura 4.15.** Vari esempi di profili di espressione ricostruiti relativi a 50 campioni-soggetti e caratterizzati da errori relativi di varia entità.
In A) I e II si hanno rispettivamente il profilo "vero" e quello ricostruito, ai quali corrisponde un errore =0.
In B) I e II si hanno rispettivamente il profilo "vero" e quello ricostruito, ai quali corrisponde un errore = 0.0672.
In C) I e II si hanno rispettivamente il profilo "vero" e quello ricostruito, ai quali corrisponde un errore = 0.3072.
In D) I e II si hanno rispettivamente il profilo "vero" e quello ricostruito, ai quali corrisponde un errore = 0.5472.
In E) I e II si hanno rispettivamente il profilo "vero" e quello ricostruito, ai quali corrisponde un errore = 0.7696.

Va inoltre osservato come l'errore non assuma qualsiasi valore all'interno del *range* [0,1], ma solo alcuni di questi valori; ciò deriva dalla definizione adottata ed è evidente osservando la distribuzione degli errori nel caso di *reordering* con un numero di campioni pari a N=50 e relativo a 100 data sets, Figura.4.16, e il relativo istogramma, entrambi riferiti a dati simulati ai quali non è stato aggiunto rumore.

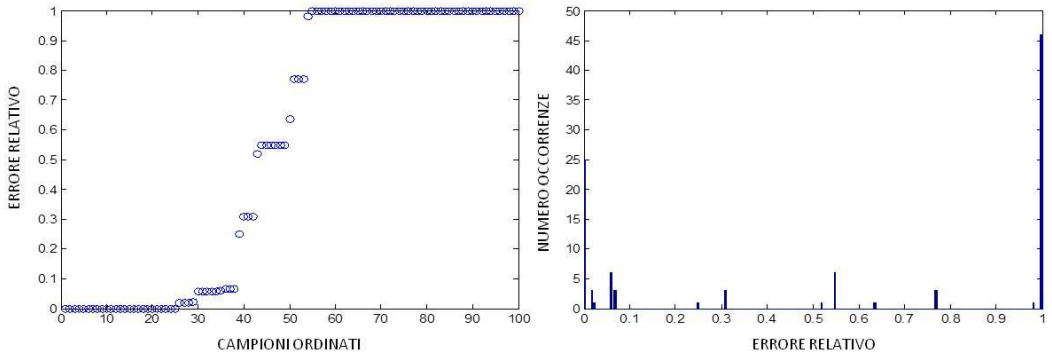



**Figura 4.16. A sinistra.** Plot degli errori relativi ordinati per valore crescente per ordinamenti ricostruiti relativi a 100 data set simulati privi di rumore. Si può notare come la distribuzione degli errori non sia continuo. **A destra.** Istogramma delle occorrenze degli errori relativi per ordinamenti ricostruiti relativi a 100 data set simulati privi di rumore.

Per via empirica si è infine determinato il valore medio degli errori corrispondenti ad un ordinamento ricostruito in maniera casuale. Per fare ciò sono stati generati un vettore ordinamento di riferimento, di lunghezza N=50, costituito da numeri interi compresi tra 1 e N senza ripetizioni in ordine casuale, e 10000 vettori di eguale lunghezza costituiti da numeri interi tra 1 e N in ordine random. Gli andamenti degli errori sono visibile in Figura 4.17 e i risultati nella tabella Tabella 4.2. Tali valori sono utili per la delimitazione di una "*random zone*", definita dai valori appartenenti all'intervallo [(0.67-0.06),(0.67+0.06)] per quanto riguarda gli errori riferiti a ordinamenti diretti e inversi, mentre dai valori appartenenti all'intervallo [(0.62-0.04),(0.062+0.04)] per quanto concerne i "*best error*". Tale "*random zone*" sarà utile termine di paragone al fine di verificare l'effettiva bontà dell'ordinamento ottenuto.

Per concludere questa parte della trattazione riguardante l'errore si riprende un concetto già espresso in precedenza, ossia il fatto che il parametro più significativo, o meglio l'unico parametro davvero significativo date le premesse (mal posizione del

problema di ricostruzione della direzionalità dell'ordinamento), è il "*best error*", cioè il minore, quindi migliore, tra lo errore diretto e inverso.

Una volta definita la formulazione dell'errore relativo come visto in precedenza verranno utilizzati come indicatori delle performance del metodo su data set estesi la precisione e la recall, tramite la quantificazione del numero di veri positivi, veri negativi, falsi positivi e falsi negativi. Precisione e *Recall* sono due classificazioni statistiche molto usate. La precisione può essere vista come una misura di esattezza o fedeltà, mentre la *recall* è una misura di completezza. In un processo di classificazione statistica, la precisione per una classe è il numero di veri positivi (il numero di oggetti etichettati correttamente come appartenenti alla classe) diviso il numero totale di elementi etichettati come appartenenti alla classe (la somma di veri positivi e falsi positivi, che sono oggetti etichettati erroneamente come appartenenti alla classe). *Recall* in questo contesto è definita come il numero di veri positivi diviso il numero totale di elementi che attualmente appartengono alla classe (per esempio la somma di veri positivi e falsi negativi, che sono oggetti che non sono stati etichettati come appartenenti alla classe ma dovrebbero esserlo).

In un processo di classificazione, un valore di precisione di 1 per la classe C significa che ogni oggetto etichettato come appartenente alla classe C al contrario non appartiene alla classe C mentre un valore di *recall* pari ad 1 significa che ogni oggetto della classe C è stato etichettato come appartenente alla classe C. In un processo di classificazione, i termini vero positivo, vero negativo, falso positivo e falso negativo sono usati per confrontare la classificazione di un oggetto (l'etichetta di classe assegnata all'oggetto da un classificatore) con la corretta classificazione desiderata (la classe a cui in realtà appartiene l'oggetto).

Precisione e recall sono definite come:

$$Precision = \frac{vero\ positivo}{vero\ positivo + falso\ negativo}$$

$$Recall = \frac{vero\ positivo}{vero\ positivo + falso\ positivo}$$

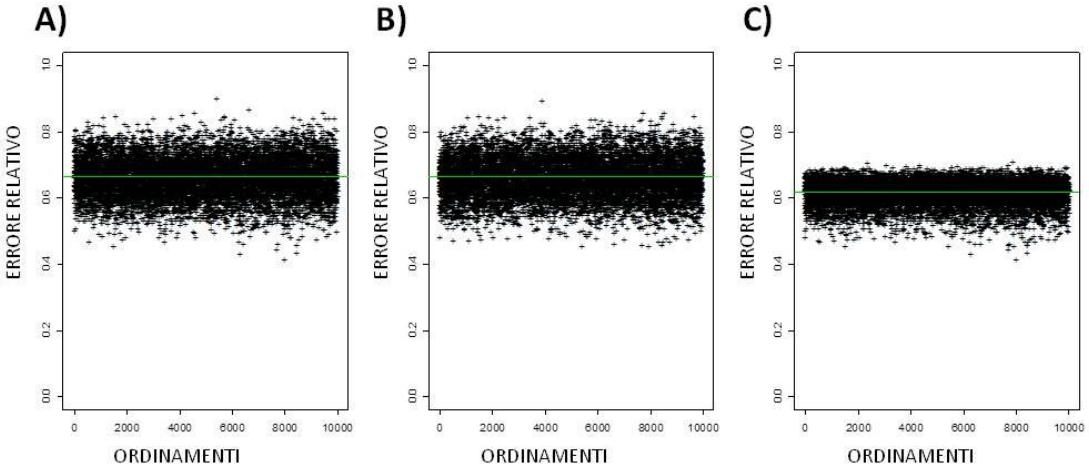



**Figura 4.17.** A) B) Errori relativi riferiti a confronto tra ordinamento di riferimento e ordinamenti casuali rispettivamente presi con verso diretto e inverso. C) Errore relativo "best error", cioè viene considerato per ogni ordinamento solo il minore tra i due errori relativi (inverso e diretto). Osservando queste figure è possibile confermare per via grafica i valori in Tabella 4.2.

| | ER diretto | ER inverso | ER best |
|---|---|---|---|
| **Media** | 0.67 | 0.67 | 0.62 |
| **Standard deviation** | 0.06 | 0.06 | 0.04 |

**Tabella 4.2.** Medie e deviazioni standard relative agli errori relativi ottenuti (diretto, inverso e best) nei 10000 ordinamenti "ricostruiti" in maniera casuale. Come prevedibile la media e la standard deviation relativa al "*best error*" è decisamente inferiore rispetto a score diretto e inverso che sono invece tra loro equivalenti in quanto a media e standard deviation.

# Appendici al Capitolo 4

## A. Scelta dei degrees of freedom (df) e ordine della derivata:

In questa Appendice si trattano alcune problematiche riguardanti l'implementazione del metodo proposto di riassetto dei blocchi: la scelta del valore da attribuire nell'utilizzo della funzione *smooth.spline* al parametro df (*the desired equivalent number of degrees of freedom*), e la scelta dell'ordine della derivata da considerare per identificare i blocchi e successivamente scegliere la combinazione winner, oltre alla scelta dei parametri da utilizzare nella fase di clustering per l'individuazione dei candidati picchi.

Si effettua come primo passo un'iniziale analisi qualitativa della questione. Come si può notare in Figura 4.A.1, aumentare il valore dei degrees of freedom per bassi livelli di rumore garantisce una più facile identificazione dei break points, ma aumentando il rumore sui dati, finisce con il diminuire di fatto l'SNR. Si può osservare come questo fenomeno peggiori considerando la derivata di secondo ordine, che risulta dunque più sensibile al rumore.

Per scegliere i valori più opportuni del *degrees of freedom* sono stati creati 100 data set simulati e 9 livelli di rumore sui dati per ognuno. I data set sono stati creati come visto nella sezione 3.1 e successivamente suddivisi in blocchi il cui ordine è stato randomizzato assieme al verso. L'ordine dei profili di espressione così ottenuto è stato ricostruito con il metodo del riassetto dei blocchi utilizzando diversi valori dei df. Oltre a variare tale valore, si è effettuato il clustering sia su break points che su break joints, (nel caso di break joint si è effettuato un clustering sia mono che bi-dimensionale), e anche sulle sola sottomatrice costituita dai soli profili differenzialmente espressi. I risultati del riordinamento con i vari setting sono stati valutati in termini di precision e recall (vedi sezione 4.4) per ogni livello di rumore sui dati. In Figura 4.A.2 e Figura 4.A.3 sono visibili i risultati in forma grafica. La recall è ottima per tutti i setting del metodo (Figura 4.A.4 e Figura 4.A.5) e ciò significa che vi sono pochi falsi negativi, e dunque il parametro decisionale diventa l'analisi della precisione. In tutti i casi si rivela essere migliore il clustering con numero cluster=3 effettuato su break point, mentre il numero di degrees of freedom adatto varia a seconda del livello di rumore sui dati Più i dati son rumorosi e più diminuisce il numero di degrees of freedom appropriato che passa da un

valore di 40 per dati privi di rumore a un valore di 20 per dati molto rumorosi. Dunque è necessaria una valutazione a priori del livello di rumore sui dati prima di settare i parametri del metodo.

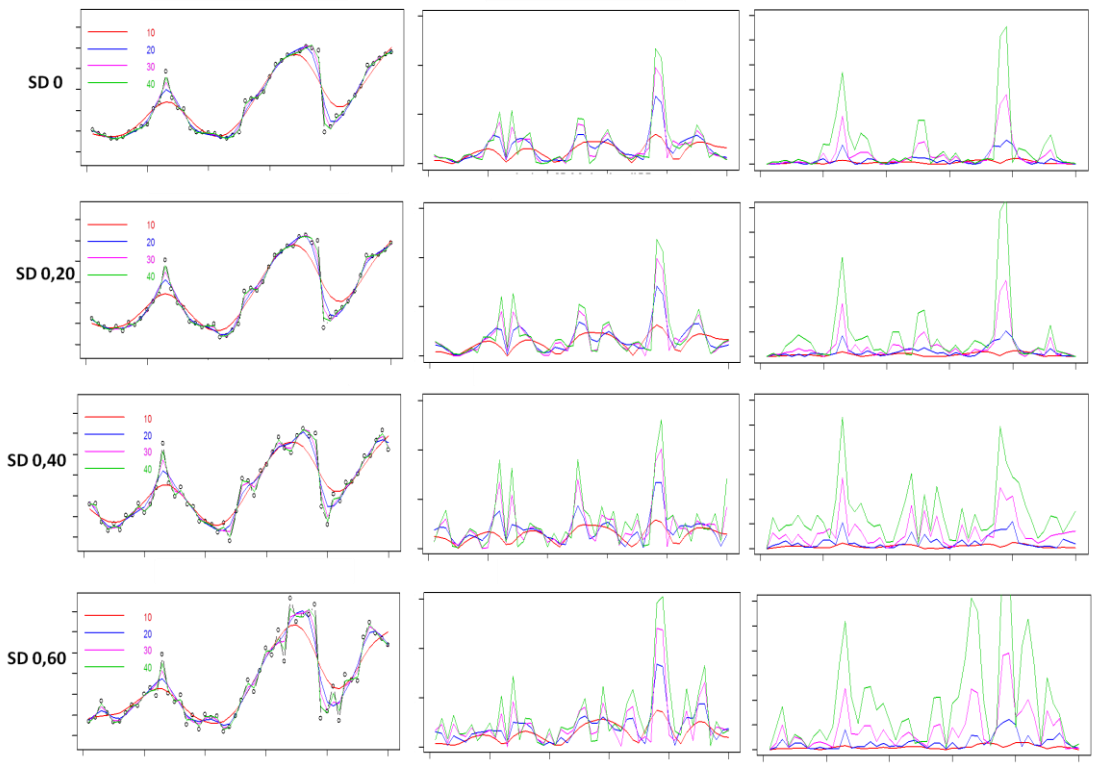



**Figura 6.A.1**. A sinistra: profilo con presenza di blocchi. Al centro: derivata prima relativa al profilo di sinistra. A destra derivata di secondo ordine del profilo di sinistra. Analisi qualitativa del rapporto tra degreees of freedom e profilo *smooth* ottenuto con le *spline* e profili della derivata prima e seconda, per diversi livelli di rumore sui dati.

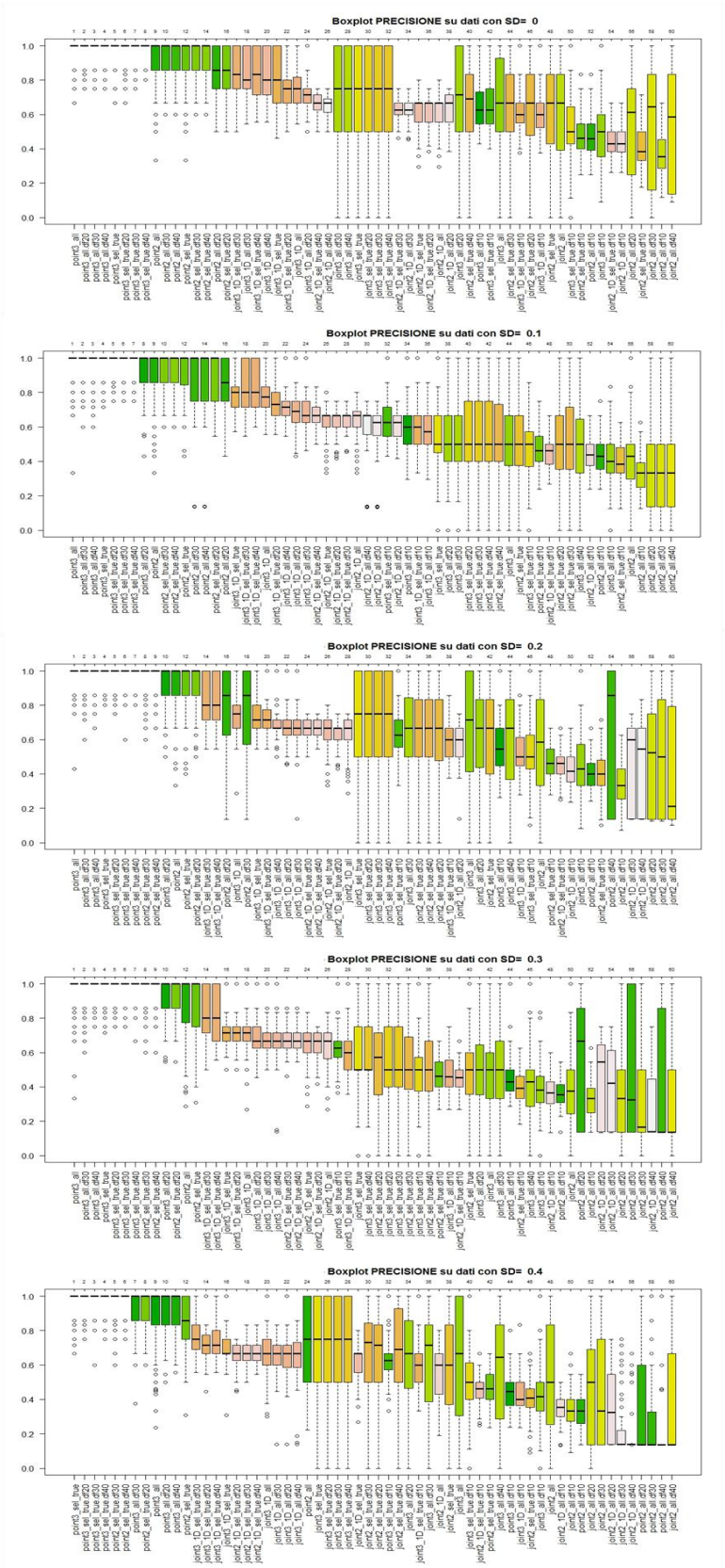



**Figura 4.A.2**. Boxplot relativi alla precisione ottenuta con i diversi valori dei parametri e con le diverse tipologie di clustering per i vari livelli di rumore sui dati (rumore a media nulla e SD=(0,0.10,0.20,0.30,0.40).

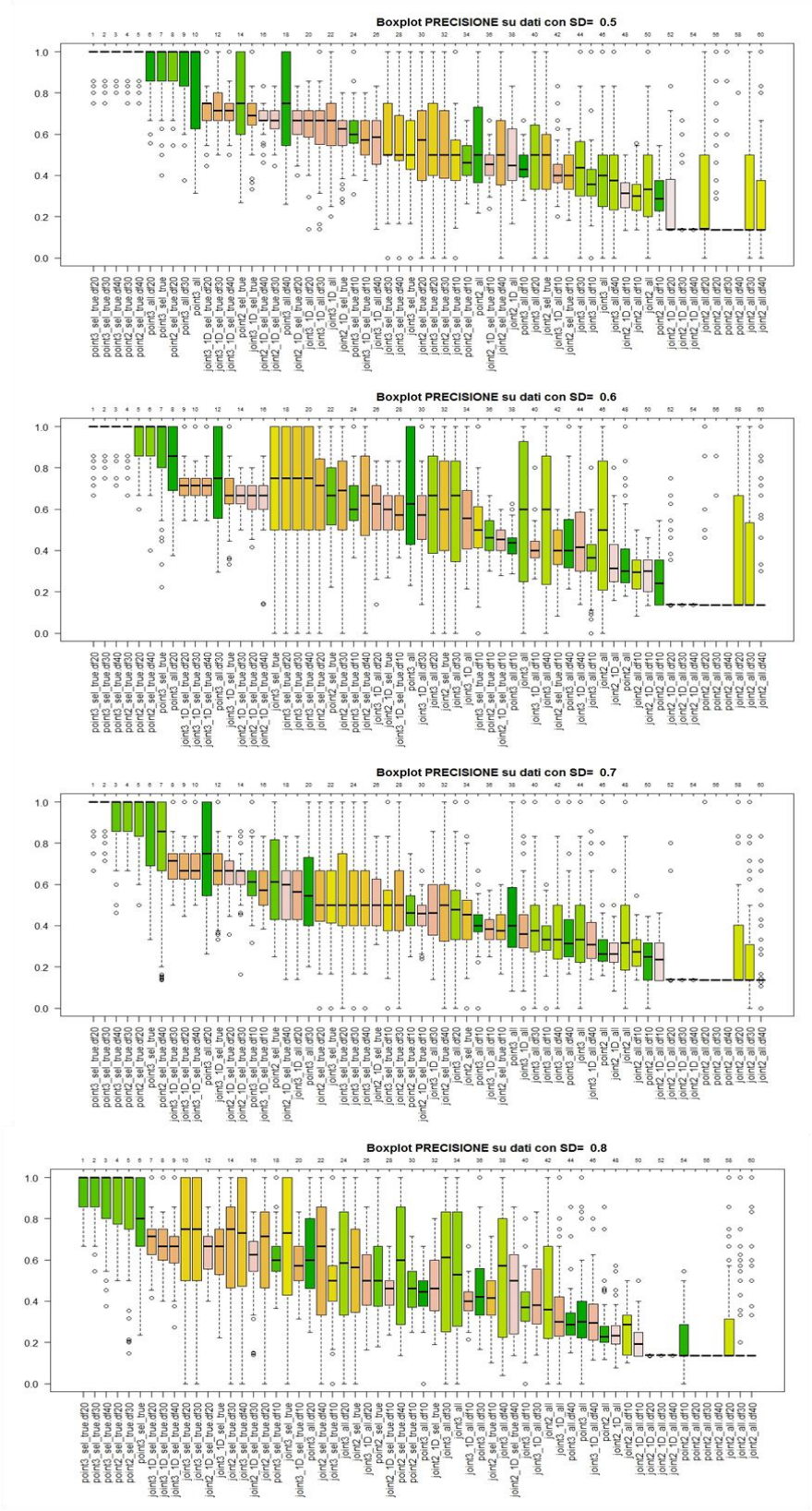



**Figura 4.A.3**. Boxplot relativi alla precisione ottenuta con i diversi valori dei parametri e con le diverse tipologie di clustering per i vari livelli di rumore sui dati (rumore a media nulla e SD=(0.50,0.60,0.70,0.80).

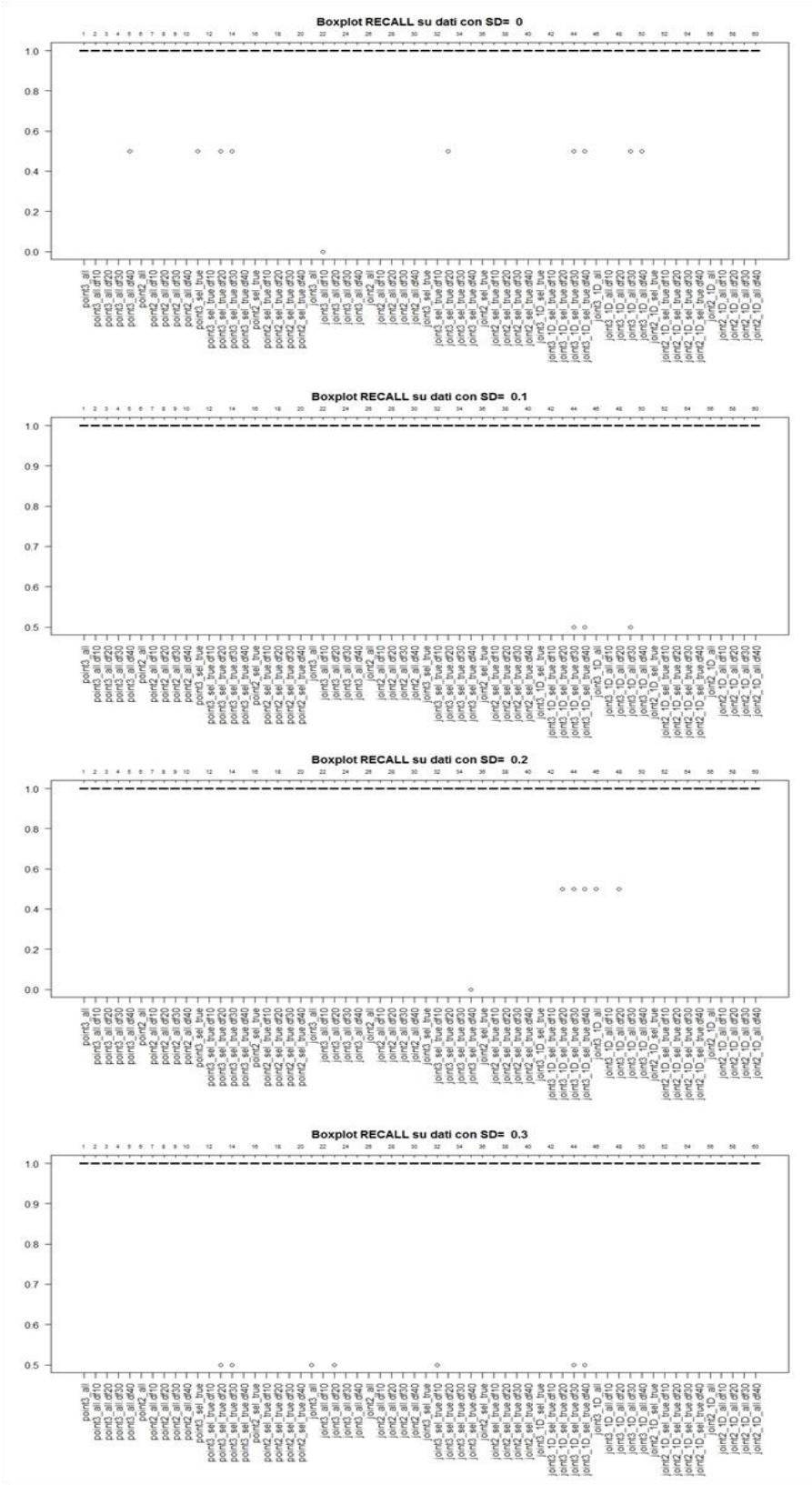



**Figura 6.A.4**. Boxplot relativi alla recall ottenuta con i diversi valori dei parametri e con le diverse tipologie di clustering per i vari livelli di rumore sui dati.

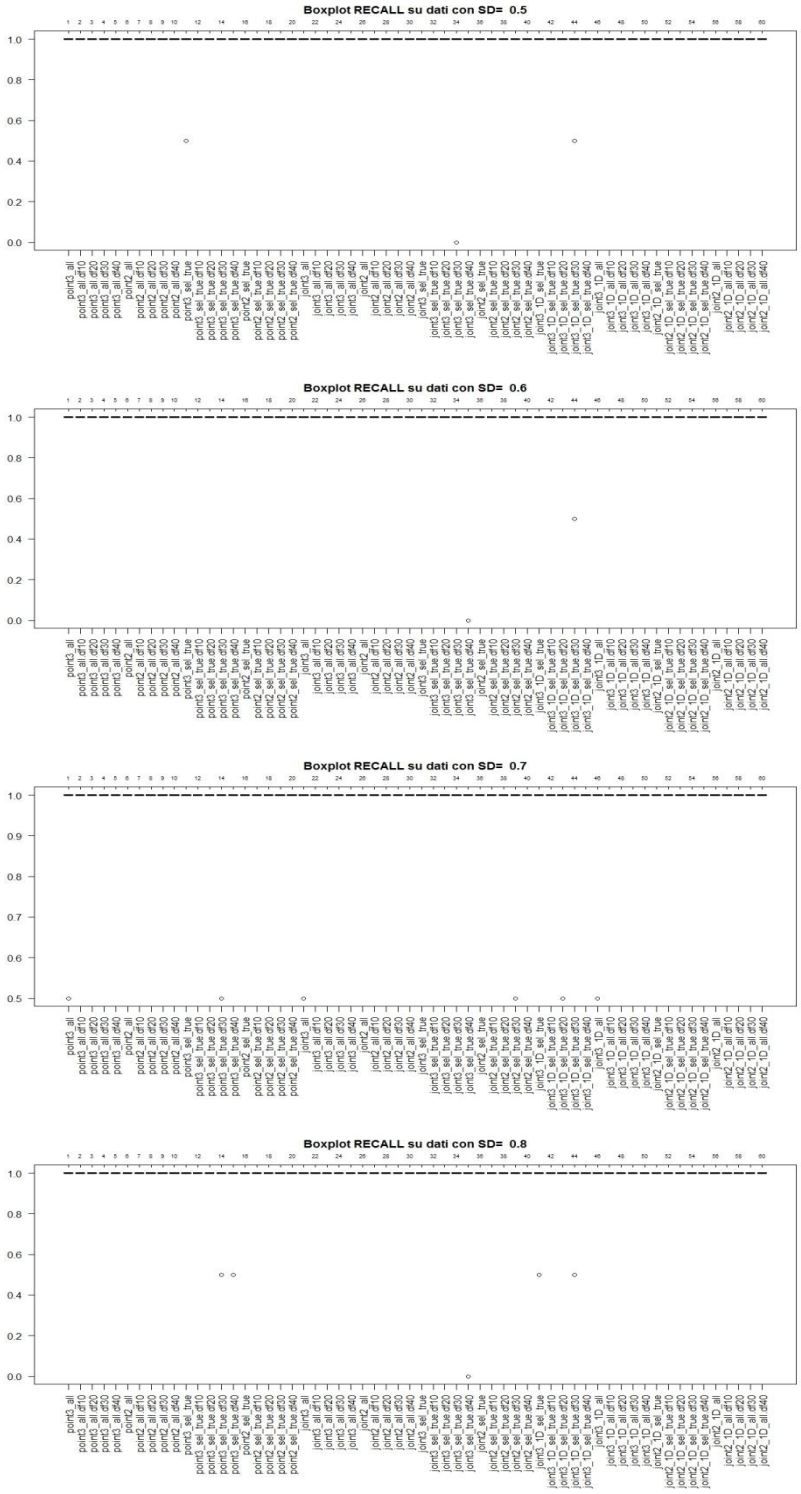



**Figura 6.A.5**. Boxplot relativi alla recall ottenuta con i diversi valori dei parametri e con le diverse tipologie di clustering per i vari livelli di rumore sui dati.

# Capitolo 5

# Risultati

## 5.1 Algoritmo originale

### Metodo deterministico

Come primo passo si è testata la capacità del metodo deterministico realizzato nello studio [1], risolto tramite un'istanza del TSP, di ricostruire correttamente l'ordinamento, utilizzando dati simulati rumorosi realizzati come descritto in 3.1.

Per ognuno dei 320 data set si è ricavato l'ordinamento deterministico ricostruito tramite la *function initOrder,* implementa in MATLAB. Dal confronto dell'ordinamento così ottenuto con l'ordinento vero, del quale chiaramente si è tenuta precendetemente memoria, si sono ricavati tre valori (come descritto nella sezione 4.4), il primo l'errore relativo riferito all'ordinamento ottenuto preso in direzione diretta, il secondo l'errore relativo preso in direzione invertita e il terzo il "*best error*", ossia l'errore minore tra i due, che come detto più volte in precedenza è il parametro di valutazione più significativo.

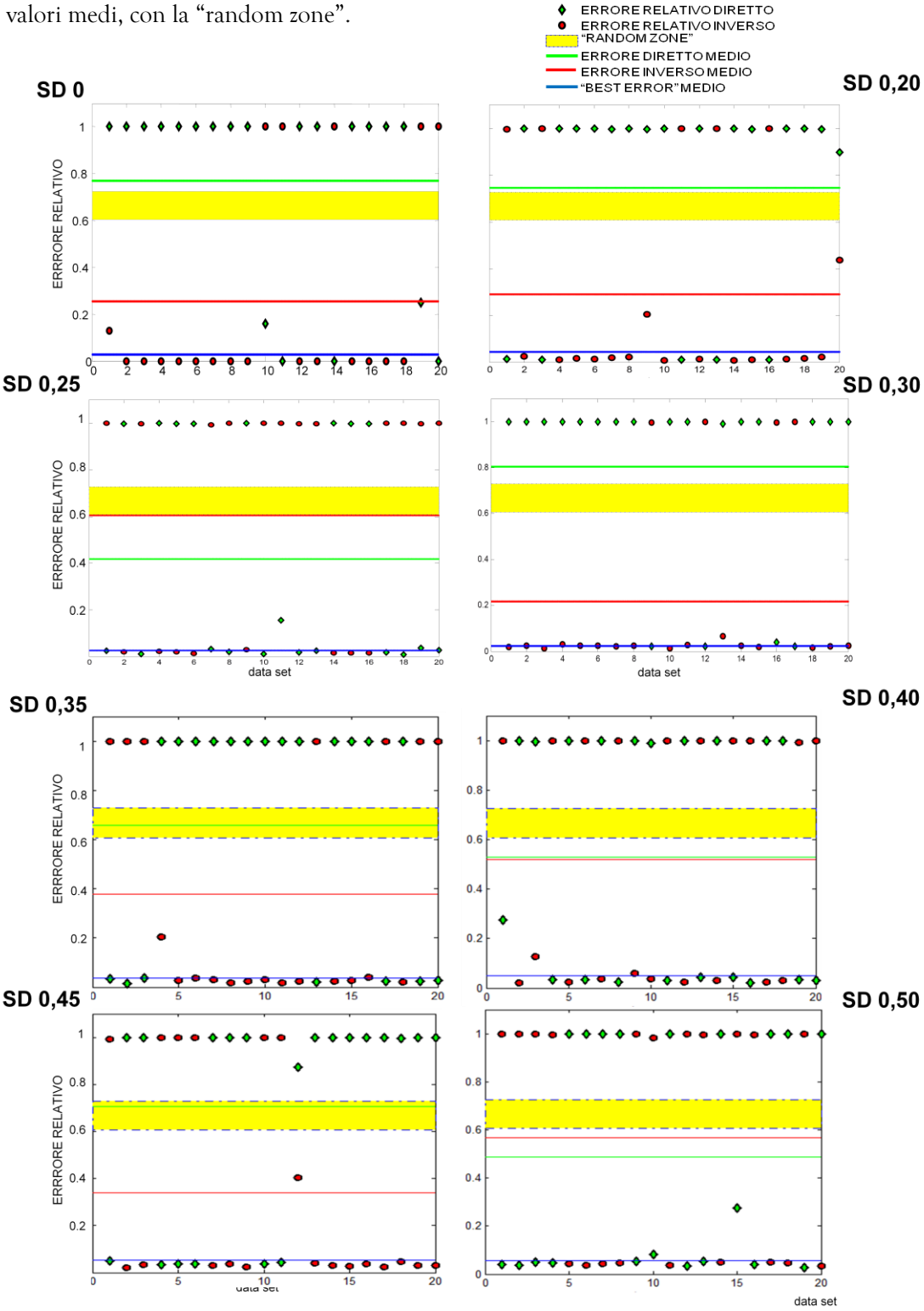
Per ognuno dei 16 livelli di rumore si è quindi calcolato il valor medio dei tre parametri (valor medio degli errori relativi diretti, degli errori relativi inversi e dei "*best error*") e le rispettive *standard deviation*. Nei grafici ottenuti si è utilizzato come termine di confronto la *"random zone"* (vedi sezione 4.4), un intervallo di valori dell'errore relativo corrispondente al confronto tra ordinamenti random.

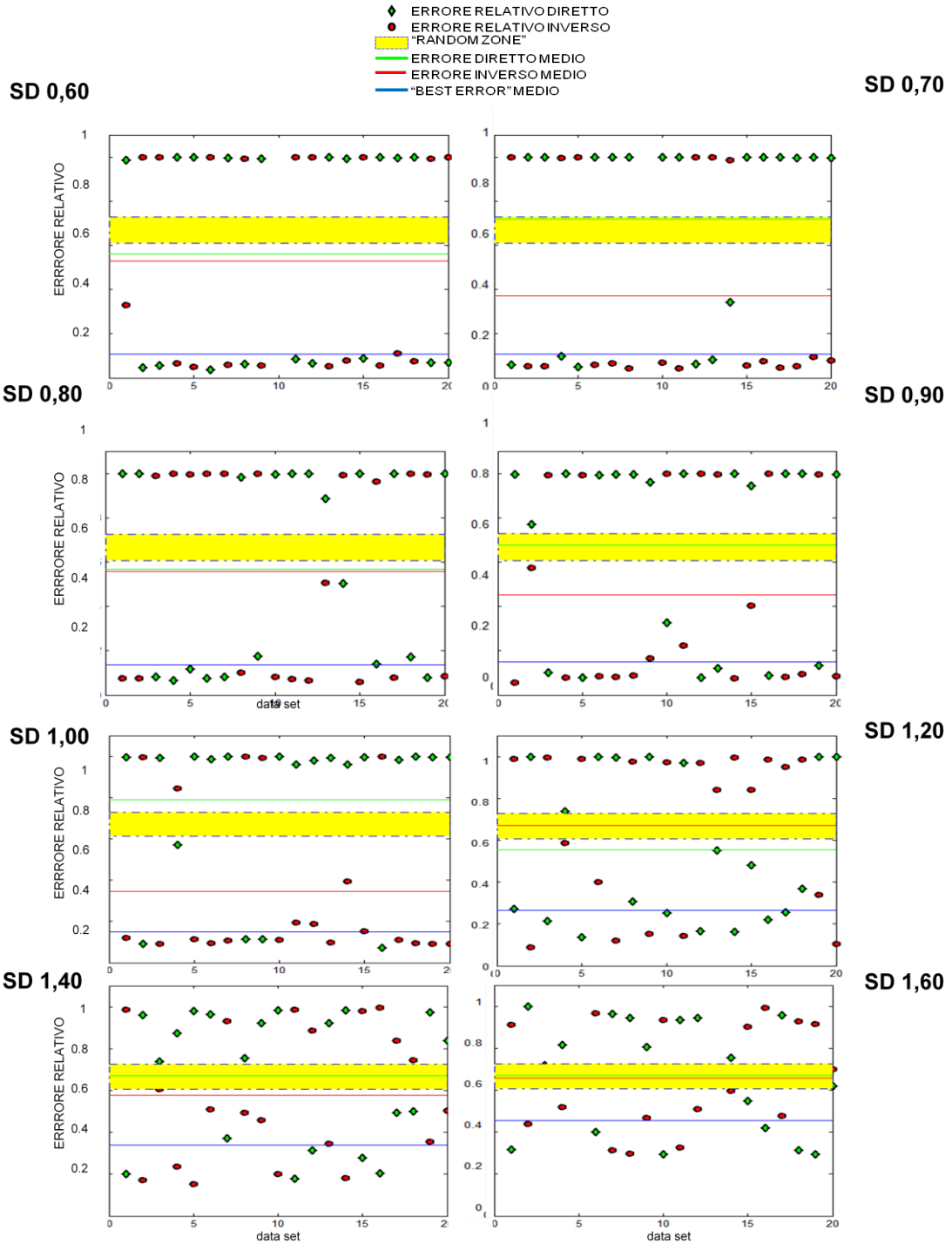
I risultati del *reordering* dei singoli data set, raggruppati per livello di rumore sui dati, sono visibili in Figura 5.2 e nella Tabella 5.1.

| | "best error" medio | Errore diretto medio | Errore inverso |
|---|---|---|---|
| **SD=0** | 0,027 | 0,77 | 0,256 |
| **SD=0,20** | 0,045 | 0,747 | 0,291 |
| **SD=0.25** | 0,027 | 0,419 | 0,607 |
| **SD=0,30** | 0,025 | 0,804 | 0,22 |
| **SD=0,35** | 0,036 | 0,66 | 0,377 |
| **SD=0,40** | 0,05 | 0,528 | 0,52 |
| **SD=0,45** | 0,053 | 0,705 | 0,341 |
| **SD=0,50** | 0,11 | 0,65 | 0,443 |
| **SD=0,60** | 0,108 | 0,561 | 0,529 |
| **SD=0,70** | 0,108 | 0,718 | 0,372 |
| **SD=0,80** | 0,135 | 0,57 | 0,557 |
| **SD=0,90** | 0,151 | 0,678 | 0,453 |
| **SD=1,00** | 0,151 | 0,79 | 0,344 |
| **SD=1,20** | 0,266 | 0,55 | 0,672 |
| **SD=1,40** | 0,338 | 0,672 | 0,578 |
| **SD=1,60** | 0,456 | 0,67 | 0,658 |

**Tabella 5.1**. È possibile vedere i valori assunti dall'errore medio diretto, inverso e del valore medio del "*best error*" per i vari livelli di rumore sui dati.

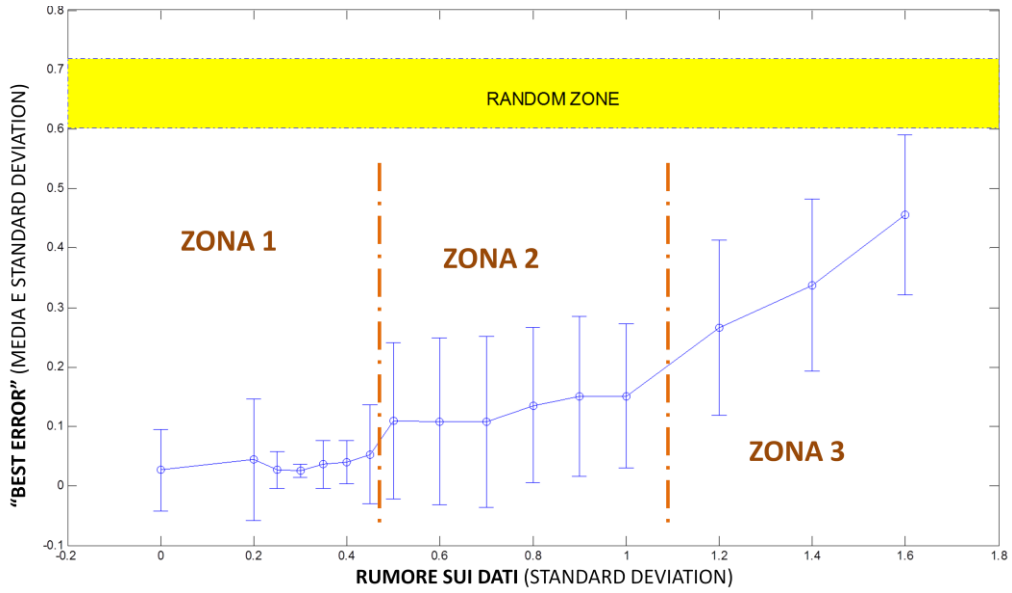
**Figura 5.2.** (Pagina corrente e pagina seguente). Grafici relativi agli errori relativi risultanti dal reordering effettuato sui 320 data set raggruppati in base al livello di rumore sui dati. Ogni grafico è considera 20 data set generati con caratteristiche analoghe. È possibile confrontare l'errore relativo diretto, inverso e il "best error", oltre ai rispettivi valori medi, con la "random zone".






Come ultima visualizzazione dei risultati ottenuti si propone un grafico, in Figura 5.3, riassuntivo, molto utile per comprendere le prestazioni del metodo deterministico. Si confrontano tra loro i valori medi dei "best error", che come detto più volte è il parametro più significativo e le rispettive standard deviation al variare della rumorosità sui dati.

**Figura 5.3.** Osservando il grafico è possibile identificare tre aree distinte, caratterizzate da dinamica dell'errore comune:
Zona 1: $0 \leq$ rumore dati $< 0.5$.
Zona 2: $0.5 \leq$ rumore dati $\leq 1$.
Zona 3: $1 <$ rumore dati.
Nella zona 1 si ottengono ordinamenti molto buoni e l'accuratezza degli ordinamenti ottenuti in tale intervallo di rumore si mantiene pressoché costante anche se la precisione è decisamente maggiore per dati affetti da rumore caratterizzato da valori della standard deviation prossimi a 0,03. Questo effetto può essere dovuto al numero esiguo di data set analizzati.
Nella zona 2 peggiorano sia accuratezza che precisione rispetto alla zona 1, mantenendosi però costanti all'interno di tale intervallo.
Nella zona 3 l'andamento dell'accuratezza degli ordinamenti ottenuti cambia comportamento e aumenta linearmente con il crescere del rumore sui dati. La precisione si mantiene pressoché costante.




Come ultimo appunto per quanto riguarda l'ordinamento deterministico, si può affermare che, soprattutto per quanto riguarda bassi livelli di rumorosità sui dati (Zona 1) gli ordinamenti associati a errore non nullo sono per lo più da imputarsi al problema dell'ordinamento in blocchi, confermando quanto visto nella sezione 1.4.

### Metodo probabilistico

Per quanto riguarda il metodo probabilistico non sono state effettuate prove altrettanto significative. Una ragione è dovuta al costo computazionale decisamente più elevato rispetto al metodo deterministico che impedisce di elaborare in tempi ragionevoli un numero elevato di data set. Una seconda ragione è che in seguito a verifiche iniziali su data set privi di rumore l'algoritmo ha condotto a pessimi risultati soprattutto se confrontato con il metodo deterministico, che si è visto essere molto buono per bassi livelli di rumore. In seguito si riporteranno i risultati di confronti incrociati tra metodo deterministico, boorstrap e metodo probabilistico, nei quali apparirà evidente la maggior affidabilità del metodo deterministico.

## 5.2 Effetto del bootstrap

Per testare la robustezza del metodo deterministico applicato al bootstrap è stata effettuata una prova sperimentale focalizzata sul riordinamento di data set simulati. Si sono riordinati i 20 data set generati per testare il metodo deterministico, vedi Figura 4.1, privi di rumore. Si è effettuato poi un confronto con il metodo deterministico e probabilistico e i risultati sono facilmente interpretabili osservando la Figura 5.4 e54.5.

**Figura 5.4.** "Best error", errori relativi diretti e inversi ottenuti riordinando i 20 data set privi di rumore con i tre metodi: deterministico, probabilistico e bootstrap. Si può notare come il metodo deterministico permetta di ottenere ordinamenti decisamente migliori e come il metodo probabilistico si riveli deludente.

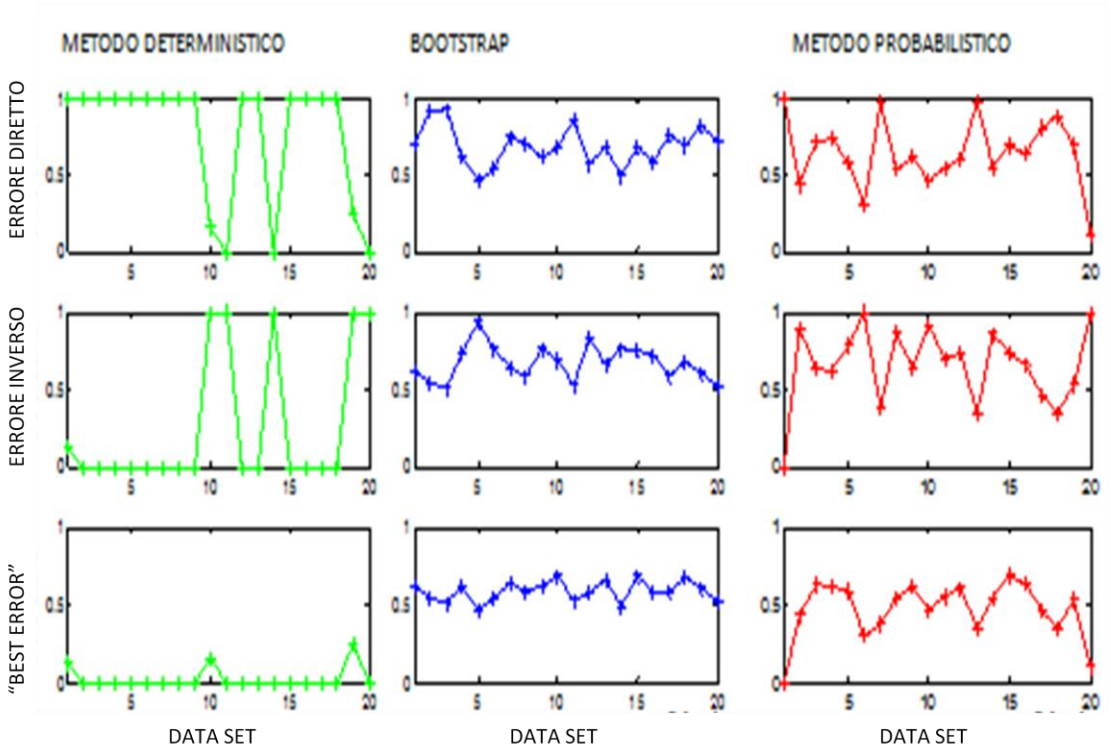



**Figura 5.5.** "*Best error*" relativo ai tre metodi.

È evidente come il *bootstrap* non permetta di migliorare la robustezza del metodo probabilistico, ma anzi ne peggiori notevolmente le performance. Inoltre come preannunciato nella sezione precedente è altrettanto evidente che il metodo probabilistico non si rivela affidabile. Per comprendere le ragioni alla base delle insoddisfacenti performance del *bootstrap* si è effettuato il riordinamento su un data set con dati non rumorosi e uno su dati affetti da rumore con SD=0,35. Le motivazioni delle cattive prestazioni fornite dall'algoritmo relativo al *bootstrap* si possono dedurre osservando le figure (Figura 5.5 e 5.6) rappresentanti il profilo assunto dall'errore relativo ("*best error*"), ottenuto confrontando gli ordinamenti ottenuti alla varie iterazioni con l'ordinamento "vero". Si può osservare come tali valori siano ad ogni iterazione molto buoni; nel caso di dati non rumorosi identici agli ottimi valori trovati con il metodo deterministico, mentre nel caso SD=0.35 di poco superiori. Mentre peggiorino al momento di fondere i 100 ordinamenti trovati in uno solo. Ciò è dovuto al fatto che il bootstrap per com'è stato concepito e realizzato non è in grado di unire due ordinamenti, magari ottimi, ma di verso opposto, senza generare un conflitto che conduce ad un ordinamento complessivo pessimo. Non è cioè in grado di discernere il verso degli ordinamenti trovati ai singoli passi. Detto in altro modo è estremamente e intrinsecamente sensibile alla mal posizione del problema di ritrovare l'ordinamento corretto anche nel verso.

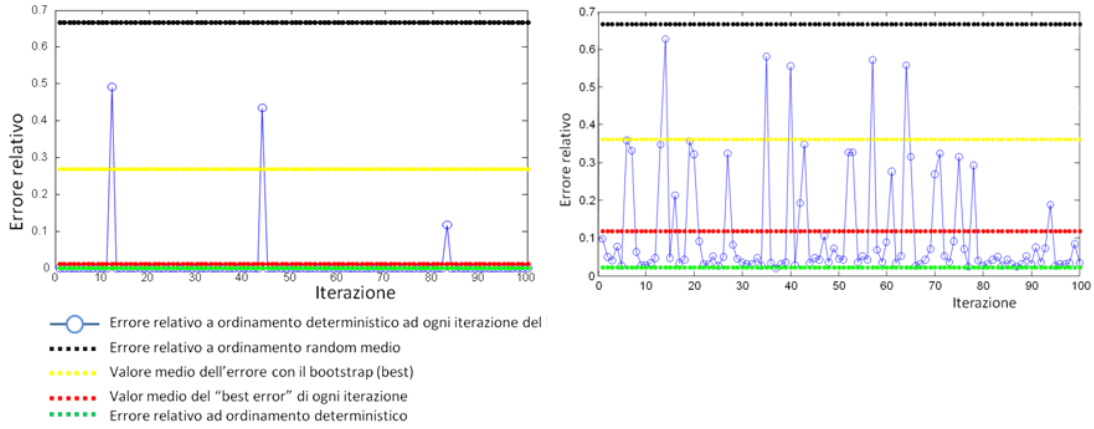



**Figura 5.6.** Grafici riguardanti i "*best error*" relativi all'ordinamento ottenuto ad ogni singola iterazione del *bootstrap*, l'errore a conclusione della procedura di *bootstrap* e successiva istanza del TSP open, confrontati con l'errore medio ottenuto in tutte le iterazioni, con l'errore relativo a ordinamenti random e con l'errore ottenuto con ordinamento deterministico. A sinistra si ha la situazione relativa al data set con dati simulati non affetti da rumore mentre a destra dati affetti da rumore caratterizzato da *standard deviation* pari a 0,35.

## 5.3 Effetto ordinamento in blocchi

Una volta generate i dati come visto in 3.1 si è applicato l'algoritmo di riordinamento deterministico, per analizzare in un secondo momento la derivata complessiva della matrice di espressione così riordinata. Una volta identificati i possibili *break points* e quindi i candidati blocchi si è proceduto alla creazione di tutte le possibili combinazioni di tali blocchi nelle due direzioni, all'analisi della loro derivata complessiva, e alla scelta della combinazione di blocchi e direzione *winner* sulla base del risultato fornito dal *clustering* dei valori della derivata (scelta della combinazione corrispondente al minimo centroide massimo, ossia alla combinazione avente il gruppo di "picchi"con valor medio minore). Sulla base della combinazione *winner* si è così riarrangiato l'ordinamento dei soggetti sintetici.

Di seguito vengono proposti i grafici rappresentativi del confronto tra i risultati ottenuti sui data set simulati con metodo deterministico e con metodo deterministico arricchito con successivo *reordering* dei blocchi.

Va notato che non sempre il riordinamento dei blocchi produce un ordinamento migliore di quello di partenze, anzi a volte lo peggiora anche per bassi livelli di rumore sui dati. Tale fenomeno è probabilmente da imputarsi alla fase di scelta dell'ordinamento dei blocchi e in particolare alla fase di clustering. Infatti non avendo imposto il numero di elementi da associare al cluster "dei picchi" tale numero varierà a discrezione del metodo scelto per il clustering e a seconda delle condizioni di partenza (nel caso del k-means), ciò falsa parzialmente la comparabilità degli ordinamenti corrispondenti alle varie combinazioni su blocchi e su direzioni. Molto probabilmente combinazioni alle quali sono associati cluster "dei picchi" con un numero maggiore di elementi, avranno un centroide con valore minore rispetto a combinazioni corrispondenti a cluster dei picchi con numero di elementi minori. Perciò alcune volte vengono scelte dal metodo combinazioni che in realtà presentano un profilo della derivata più irregolare di altre e che quindi conducono a ordinamenti errati.

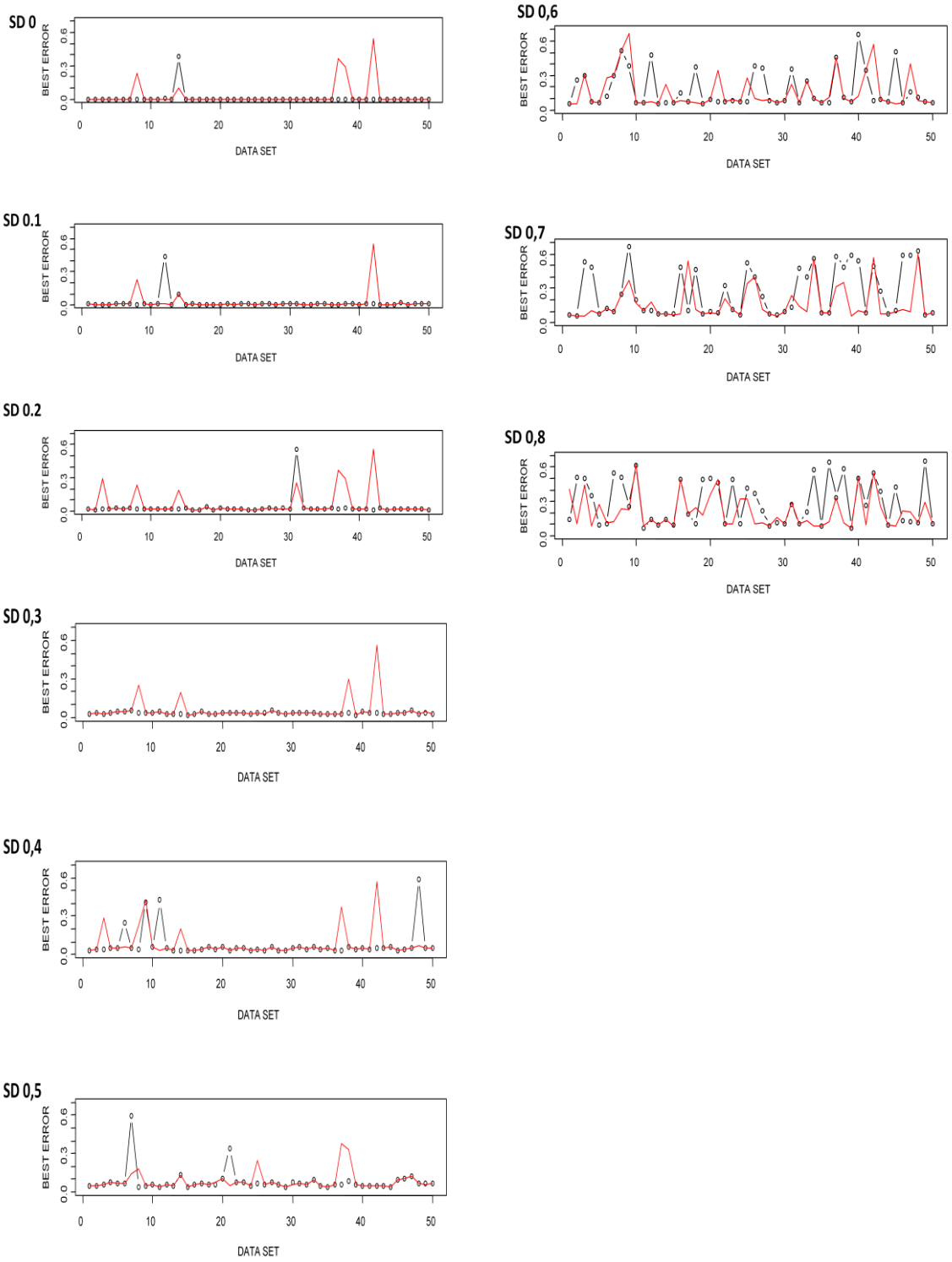



**Figura 5.7.** Errori relativi ottenuti con metodo deterministico (linea rossa) e con metodo deterministico con successivo block reordering (linea nera) , per i data set accumunati da stesse tipologie di rumore. Si può notare un aspetto che viene confermato dalle successive figure, cioè che per bassi livelli di rumore sui dati (con SD minore di 0.5) il metodo di riordinamento deterministico con block reordering in effetti riduce il numero di riordinamenti errati.

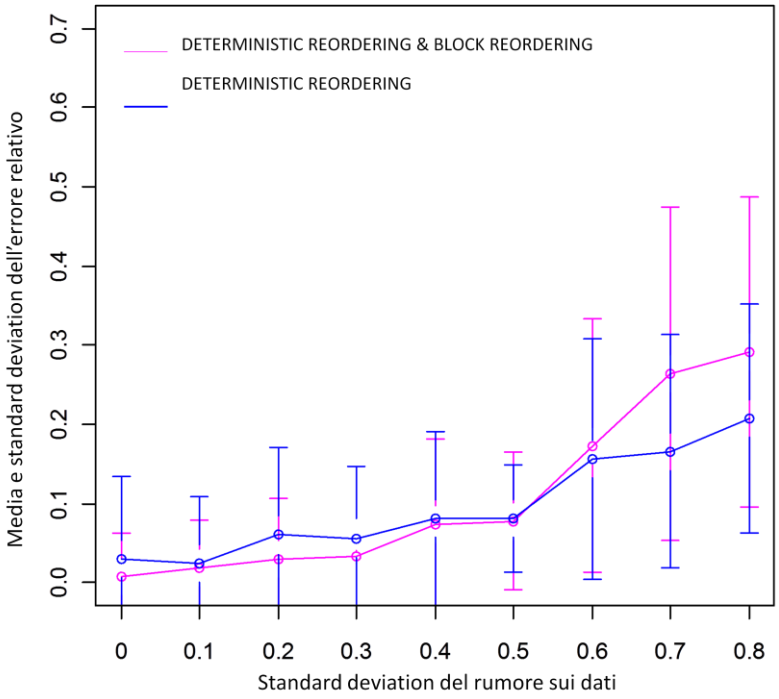



**Figura 5.8.** Questo grafico riassume in maniera sintetica, ma molto eloquente il risultato del test di simulazione. È infatti evidente come il metodo con block reordering migliori il metodo deterministico su dati poco o mediamente rumosi (SD <0.5), mentre risulti peggiore in dati affetti da alto livello di rumore.

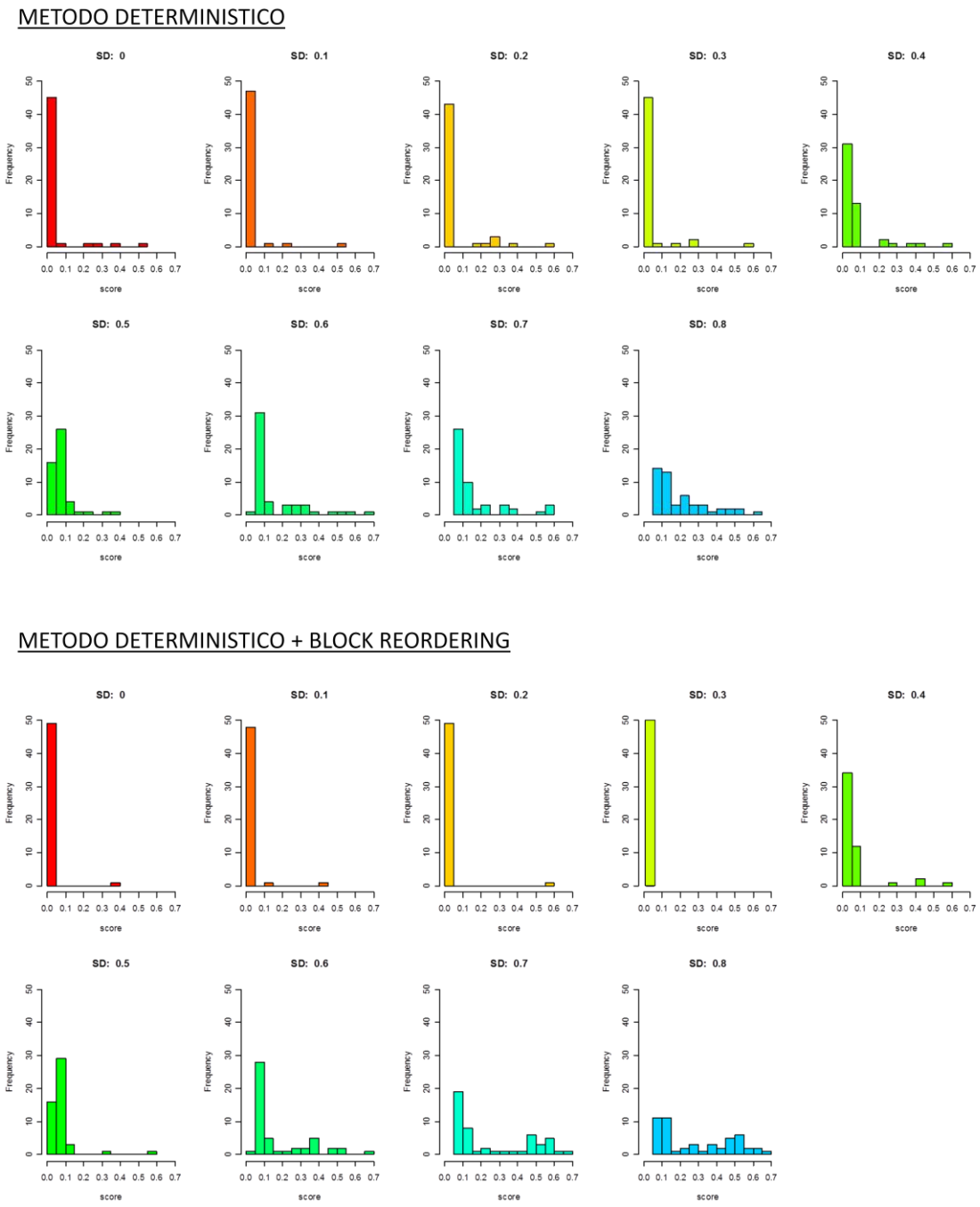



**Figura 5.10**. Istogrammi della distribuzione dei "*best error*" per ogni livello di rumore. Notare che fino al rumore con SD pari a 0.5 il metodo con *block reordering* tende a "spostare" la distribuzione del numero di occorrenze verso errori di entità minore. Cioè migliorare l'ordinamento

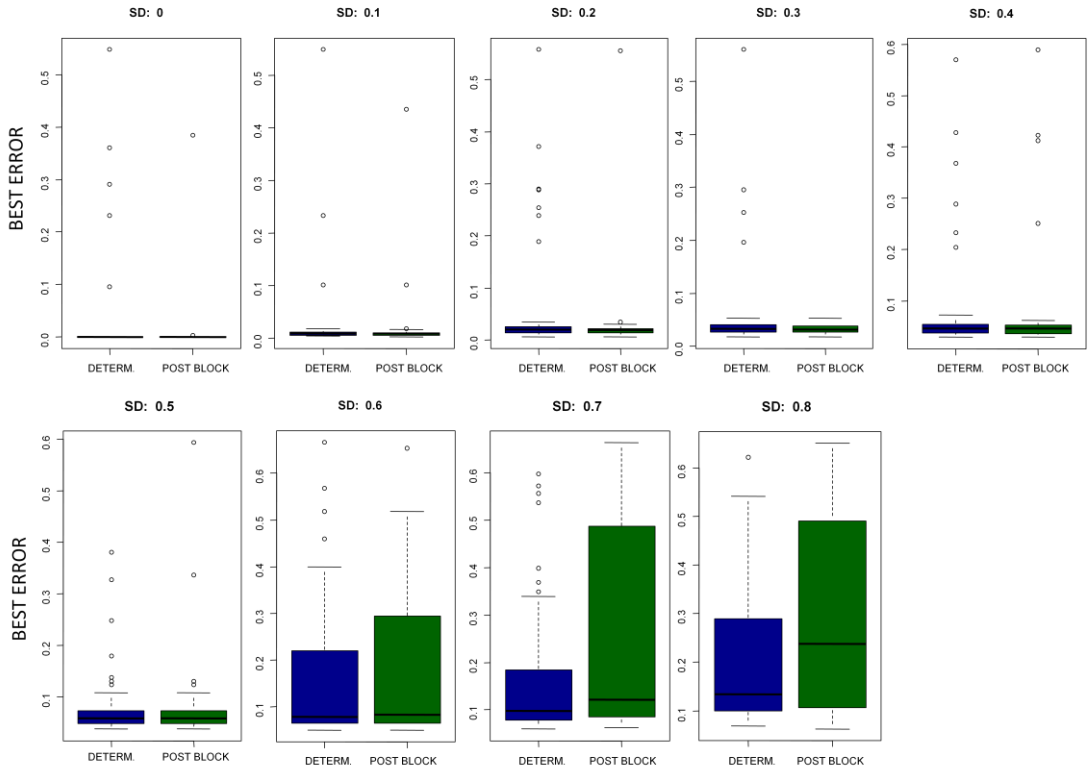



**Figura 5.11.** Confronto tra boxplot della distribuzione degli errori relativi riferiti a metodo deterministico e deterministico con block reordering. Una volta ancora è evidente che per data set poco rumorosi il metodo con il riordinamento dei blocchi migliora le performance dell'algoritmo.

# Capitolo 6

# Applicazione a dati reali

### Dati dinamici

Considerato il data set ottenuto come visto in 3.3 si è effettuata un'operazione di *shuffle* sulle 9 colonne della matrice di espressione, randomizzando di fatto l'ordine dei campioni temporali. Sulla matrice di espressione "statica" così ottenuta è stato effettuato il riordinamento con il metodo deterministico e su tale ordinamento si è svolta l'analisi delle derivate e il riordinamento dei blocchi, come visto nel Capitolo 4. Una volta ottenuto l'ordinamento deterministico e l'ordinamento risultante dal successivo riassetto dei blocchi, si sono analizzati graficamente i profili delle derivate complessive relative alle rispettive matrici dei profili di espressione, confrontandoli con il profilo della derivata complessiva della matrice di espressione originaria (la matrice "statica"). Le conclusioni tratte da tale osservazione sono state validate tramite il successivo confronto con l'ordinamento "vero", cioè l'ordinamento corrispondente alla matrice dei dati dinamici, chiaramente noto.

Per quanto riguarda l'implementazione della parte di codice relativa alla scelta della combinazione dei blocchi migliori, dato il numero molto basso di campioni e di possibili blocchi, si è sostituito il clustering con un'operazione di sorting dei valori corrispondenti alle derivate prime complessive associate alle matrici di espressione riordinate secondo le varie combinazioni possibili. Si è poi scelta la combinazione avente minor valor medio dei primi (2) valori dei vettori così ricavati.

Dopo aver effettuato questi passaggi l'ordinamento risultante dal riassetto dei blocchi ricavato è risultato essere identico all'ordinamento ricavato in precedenza con il metodo deterministico e dunque si può affermare che non si è rilevata la presenza di blocchi.

Osservando la figure con i profili delle derivate di primo e secondo grado (Figura 6.1) si può notare come l'ordinamento deterministico produca una derivata complessiva molto più regolare rispetto all'ordinamento random e con valor medio minore, sintomo di una pendenza mediamente minore (quindi indice di maggior regolarità dei profili). Ciò fornisce una prima conferma della bontà dell'ordinamento ottenuto.

Dal confronto con l'ordinamento "vero", ovvero l'ordine relativo i profili dinamici originari, si evince che l'ordinamento ricostruito è in effetti l'ordine corretto e dunque sia il metodo deterministico che il riassetto dei blocchi hanno dato risultati convergenti e soddisfacenti.

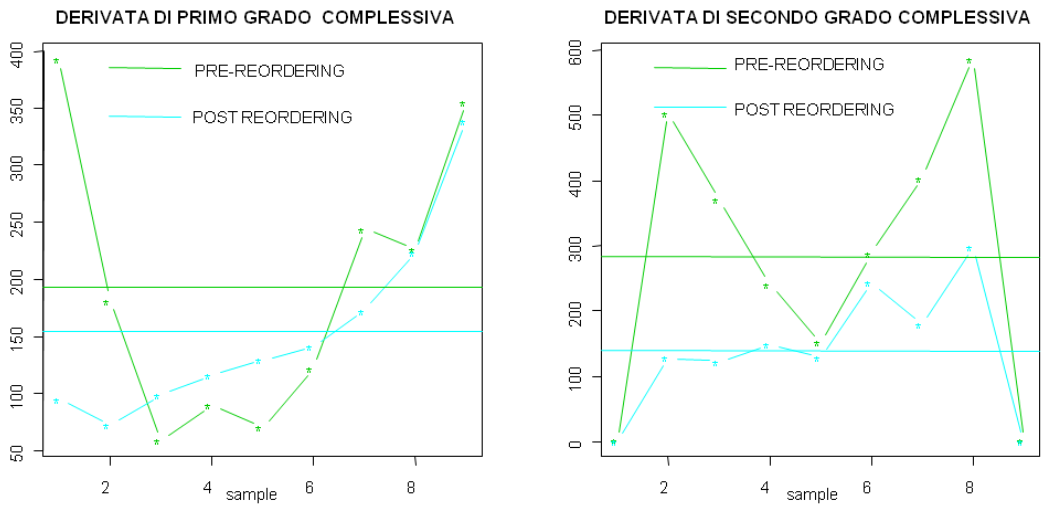



**Figura 6.1**.Grafici riguardanti la derivata complessiva prima e dopo la procedura di riordinamento deterministico. A sinistra si ha un confronto tra le derivate di ordine 1. A destra un confronto tra le derivate di ordine 2. Si può notare come in entrambi i casi la derivata riferita alla matrice di espressione riordinata abbia media minore e come (soprattutto nel grafico di sinistra) sia decisamente più regolare.

**Dati statici**

Come ultimo passo dello studio svolto si è effettuata una procedura di riordinamento su dati statici di espressione genica, ottenuti come visto in 3.2, condotta con l'ausilio del continuo confronto con il contenuto informativo riguardante i marker prognostici IgVh e ZAP-70.

Innanzitutto si è effettuato il riordinamento dei soggetti con metodo deterministico, probabilistico e con le rispettive analisi dei blocchi, su data set di dimensioni variabili, ottenuti dalla matrice di espressione statica iniziale effettuando due diverse procedure di selezione. Il fine di questa parte dell'elaborazione è stato quello di ottenere una conferma clinica della bontà del metodo di ordinamento con il successivo riassetto dei blocchi, e scegliere il metodo di riordinamento che concordi maggiormente con i parametri prognostici utilizzati nella pratica clinica, raccolti dai soggetti al momento della diagnosi della malattia. Una volta scelto il metodo, si è effettuata una successiva fase di clustering sui profili dei geni riordinati. Per questa elaborazione si è ricorso al clustering k-means e gerarchico divisivo (implementato con la funzione DIANA del pacchetto *cluster* di R).

I valori numerici presenti in Tabella 6.1 corrispondono al numero dei cambiamenti di stato che si sono ottenuti per i vari parametri prognostici (IgVh, ZAP-70) e per le classi di appartenenza, nei 112 soggetti riordinati con le varie metodiche. I soggetti sono stati riordinati considerando diversi sottoinsiemi del data set riferiti a 678 e 97 probe set, oltre al data set complessivo di 1970 probe set. Va ricordato che tra i vari parametri prognostici considerati, le mutazioni nei geni che codificano per IgVh e l'attribuzione delle classi 1-2 rivestono una maggior significatività clinica.

Considerando il primo data set (relativo a 1970 probe set) per quanto riguarda l'IgVh si hanno prestazioni identiche per tutti i metodi mentre se si osserva la suddivisione in classi 1-2 si nota che al riordinamento ottenuto con il metodo deterministico con riassetto dei blocchi corrisponde un numero di cambi di stato minore (8). Tale scelta viene confermata anche dagli altri due parametri. Osservando il secondo data set (ottenuto da una prima grossolana selezione dei geni del primo data set) si nota come il metodo probabilistico sembri migliore rispetto al deterministico mentre nel terzo data set le prestazioni osservate tramite il numero di cambi di stato dei parametri sembrano essere all'incirca identiche per tutti i metodi.

Osservando i cambi di stato totali sugli ordinamenti prima e dopo il riassetto dei blocchi si evince che il riassetto ha l'effetto di ridurli e dunque la distribuzione dei parametri più significativi (*) dal punto di vista clinico supporta tale metodologia.

Sia il metodo deterministico che probabilistico forniscono ordinamenti ai quali corrispondono un basso numero di cambi di stato per quanto riguarda i due parametri principali: l'IgVh e l'attribuzione delle classi 1-2.

| | 1970 | | | | 678 | | | | 97 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **reordering** | **P** | **D** | **PB** | **DB** | **P** | **D** | **P B** | **D B** | **P** | **D** | **P B** | **D B** |
| ***IgVh** | 8 | 8 | 8 | 8 | 7 | 8 | 6 | 8 | 7 | 7 | 7 | 7 |
| **ZAP-70** | 39 | 35 | 39 | 34 | 35 | 40 | 33 | 39 | 40 | 40 | 40 | 41 |
| **Classi 1-2-3** | 41 | 40 | 41 | 39 | 36 | 42 | 35 | 41 | 42 | 44 | 42 | 45 |
| ***Classi 1-2** | 9 | 10 | 9 | 8 | 5 | 10 | 5 | 9 | 5 | 6 | 5 | 5 |
| **Tot su *** | 35 | | 33 | | 30 | | 28 | | 25 | | 24 | |

**Cambi di stato totali su D e P: 90**

**Cambi di stato totali su D e P con RIASSETTO DEI BLOCCHI: 85**

**Tabella 1**. In questa tabella sono presenti il numero di cambi di stato relativi parametri: IgVh, ZAP-70, Classi 1-2-3 e Classi 1-2. Tali valori sono stati calcolati sugli ordinamenti ottenuti con il metodo deterministico (D), probabilistico (P) e i rispettivi riordinamenti dei blocchi (D B) e (P B). I data considerati corrispondono al data set originale (1970 profili), e i data set ricavati tramite una selezione dei profili del data set originario, il primo con 678 profili, il secondo con 97. Nell'ultima riga è presente un confronto tra i valori prima e dopo il riassetto dei blocchi. È possibile notare come sia sempre inferiore il numero dei cambi di stato relativo ai metodo con riassetto dei blocchi rispetto ai metodi deterministico e probabilistico originari.

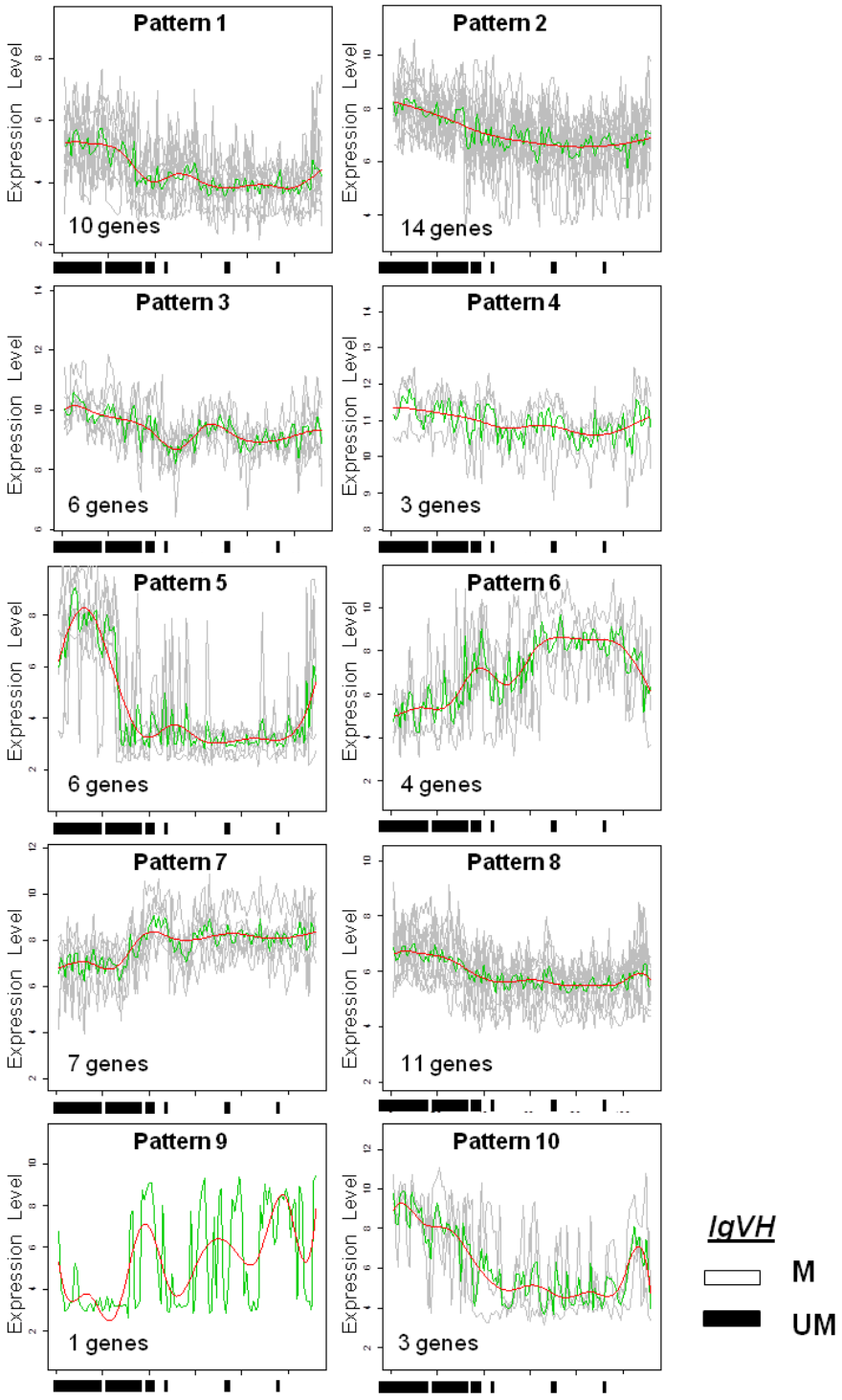



**Figura 6.2.** Rappresentazione della suddivisione in cluster dei profili temporali riordinati con il metodo deterministico e il successivo riassetto dei blocchi. ottenuta con il clustering gerarchico divisivo. Sotto ad ogni grafico è presente la distribuzione del parametro clinico relativo ai geni che codificano per IgVh nei soggetti riordinati. È' possibile notare la presenza di una separazione tra soggetti con geni mutati (M) e non mutati (UM). Ciò può essere visto come una concordanza tra la dinamica temporale ottenuta e le valutazioni cliniche.

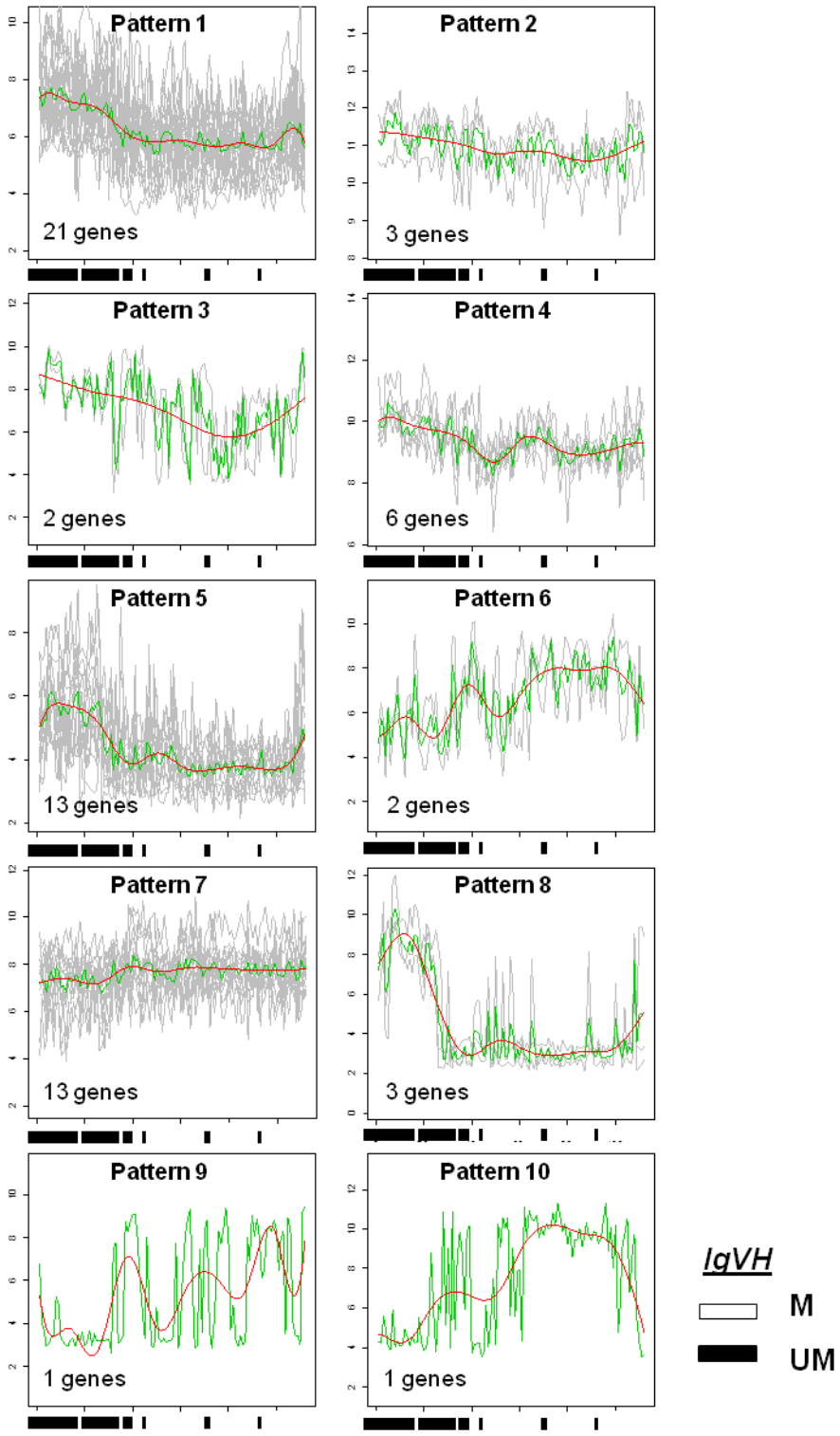



**Figura 6.3.** Rappresentazione della suddivisione in cluster dei profili temporali riordinati con il metodo deterministico e il successivo riassetto dei blocchi. ottenuta con il clustering k-means. Sotto ad ogni grafico è presente la distribuzione del parametro clinico relativo ai geni che codificano per IgVh nei soggetti riordinati.

# Capitolo 7

# Conclusioni e sviluppi futuri

La prima parte del lavoro svolto è stata dedicata alla comprensione delle possibilità e dei limiti del metodo di riordinamento temporale di dati statici di espressione proposto da Anupam Gupta e Z. Bar-Joseph. Si è voluta integrare la trattazione fornita nel loro lavoro con un'ampia validazione su dati statici simulati . I data set simulati generati nel lavoro di riferimento erano composti interamente da geni creati come significativi (ricavati da profili temporali), mentre è più realistico considerare data set nei quali una parte minoritaria sia quella dei geni differenzialmente espressi e la gran parte dei geni sia fondamentalmente rumore.

Anupam Gupta e Z. Bar-Joesph propongono due versioni: deterministica e probabilistica. Per entrambe sono stati effettuati dei test su data set simulati non affetti da rumore e su data set affetti da basso livello di rumore (a media nulla e standard deviation pari a 0.35). Dal momento che i risultati ottenuti si sono rivelati decisamente migliori per il metodo deterministico si è proseguita la sperimentazione tralasciando il metodo probabilistico.

I test su dati simulati relativi al metodo deterministico hanno evidenziato come maggior fonte d'errore la presenza di blocchi negli ordinamenti ricavati. In particolare ci si è focalizzati sulla propensione dell'algoritmo, messo a punto dagli autori, a ricostruire ordinamenti con la presenza di blocchi ordinati al loro interno correttamente, ma permutati tra loro e in verso di percorrenza sia diretto che invertito.

Da questo punto dello studio in avanti ci si è dedicati alla creazione di un metodo in grado di limitare questo fenomeno. Le vie percorse sono state due: l'utilizzo del bootstrap e il riassetto dei blocchi tramite analisi della derivata assoluta complessiva.

Nel primo caso i risultati ottenuti non sono stati soddisfacenti in quanto hanno peggiorato le prestazioni ottenute con il solo metodo deterministico, mentre nel secondo caso si sono ottenuti risultati incoraggianti. L'individuazione dei blocchi, ottenuta mediante la rilevazione dei picchi presenti sulla derivata assoluta estesa a tutti i profili del data set, e il successivo riordinamento, hanno permesso (per livelli di rumore sui dati non eccessivi, SD<0.5) di rimediare alla presenza di blocchi e migliorare dunque l'ordinamento ricavato. In alcuni casi ciò non avviene, soprattutto per livelli di rumore sui dati più elevato (SD>0.5). Ciò è imputabile principalmente alla fase di calcolo della derivata relativa ad ogni combinazione dei blocchi, che spesso non conduce ad un'individuazione corretta e ripetibile dei picchi, e che si ripercuote sulla fase di confronto tra le diverse combinazioni per la scelta della combinazione winner.

In questo studio si è dimostrato come sia possibile porre rimedio, anche se solo parzialmente, ad alcune delle lacune evidenziate dal metodo di riordinamento temporale di dati statici di espressione proposto da Anupam Gupta e Z. Bar-Joseph.

I possibili sviluppi futuri riguardano vari aspetti. Si può pensare di creare un algoritmo in grado di contemplare l'integrazione di informazione a priori su profili temporali di geni noti (relativi all'evoluzione di un particolare evento). In tal modo si compenserebbe la mal posizione del problema di ricerca del verso di percorrenza corretto dell'ordinamento. Per comprendere meglio si pensi ad esempio di aver ricostruito, con i passaggi visti nel presente studio, un ordinamento a partire da dati statici di espressione. Per confermare tale ordinamento o la sua versione invertita, basterebbe confrontare il profilo di un gene con profilo di espressione noto (up-down regolato) con il profilo dello stesso gene nell'ordinamento ricavato. Si può inoltre rivedere il metodo bootstrap per migliorare la precisione dell'ordinamento. Una via possibile è quella di effettuare le iterazioni non più sui geni, come fatto in questo lavoro, ma sui soggetti. Ossia ricampionando ad ogni iterazione le colonne della matrice di espressione e ottenendo quindi B ordinamenti ( B = numero iterazioni). Per evitare il problema della presenza di blocchi è possibile effettuare il riassetto dei blocchi per ognuno dei B ordinamenti ottenendo un'altra serie di B ordinamenti con i blocchi riordinati. Una volta effettuati tali passaggi è possibile pensare di integrare tra loro i vari ordinamenti effettuando un'operazione di clustering sugli ordinamenti. Cioè creare M sottoinsiemi (M<B) composti da ordinamenti simili tra loro, cioè con errore relativo

basso. Si può proseguire ottenendo per ogni cluster un unico ordinamento dei soggetti, che si può supporre essere più robusto dei singoli ordinamenti presenti nel cluster, ad esempio ponendo il quesito come istanza del problema del commesso viaggiatore (TSP). Una volta effettuati questi passaggi si può procedere confrontando ogni ordinamento rappresentativo di un cluster con tutti gli altri ordinamenti nelle loro versioni inverse, e nel caso in cui si riscontri una notevole somiglianza, unire i rispettivi cluster avendo l'accortezza di invertire il verso di uno dei due. Per evitare di fondere ordinamenti conflittuali è opportuno far in modo che la varianza all'interno dei cluster sia sempre inferiore alla varianza tra cluster.

Si può pensare di utilizzare una procedura che iterativamente effettui selezione e riordinamento. È verosimile infatti aspettarsi che un procedimento di riordinamento su un data set esteso, nel quale molti dei profili corrispondano di fatto a rumore, conduca a ordinamenti complessivamente scorretti, ma che localmente, ossia per gruppi di pochi campioni, riflettano delle dinamiche temporali significative. La matrice risultante da tale ordinamento localmente corretto può dunque venire utilizzata per una prima procedura di blanda selezione dei geni differenzialmente espressi (FDR alta), assicurandoci così di non commettere dei falsi negativi. Effettuando ora un nuovo riordinamento sulla matrice dei geni selezionati è lecito attendersi, dato il minor apporto rumoroso, di ottenere un ordinamento migliore rispetto al precedente, ossia con una minor suddivisione in blocchi, i quali chiaramente avranno dimensioni maggiori. Una seconda procedura di selezione su tali dati condurrà probabilmente a un numero di falsi negativi migliore rispetto ad una procedura di selezione operata sulla matrice riordinata come al passo precedente. Reiterando selezione e riordinamento dei blocchi fino al numero di geni desiderato ci si può aspettare di ottenere risultati migliori sia sulla selezione che sul riordinamento dei soggetti.

# RIFERIMENTI BIBLIOGRAFICI

[1] Anupam Gupta e Z. Bar-Joseph, "Extracting Dynamics from Static Cancer Expression Data", *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 5, no. 2, 2008.

[2] T.R. Golub et al., "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring", *Science*, vol. 286, pp. 531-537, 1999.

[3] A.C. Gustafsson et al., "Global Gene Expression Analysis in Time Series Following N-Acetyl L-Cysteine Induced Epithelial Differentiation of Human Normal and Cancer Cells in Vitro", *BMC Cancer*, vol. 5, p. 75, 2005.

[4] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, and M.B. Eisen, "Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes", *Molecular Biology of the Cell*, vol. 11, no. 12, pp. 4241-4257, 2000.

[5] G.J. Nau et al., "Human Macrophage Activation Programs induced by Bacterial Pathogens", *Proc. Nat'l Academy of Sciences*, vol. 99, pp. 1503-1508, 2002.

[6] Z. Bar-Joseph, E. Demaine, D. Gifford, A. Hamel, N. Srebro, and T. Jaakkola, "k-Ary Clustering with Optimal Leaf Ordering for Gene Expression Data", *Bioinformatics*, vol. 19, pp. 1070-1078, 2003.

[7] N. Kaminski and Z. Bar-Joseph, "A Patient-Gene Model for Temporal Expression Profiles in Clinical studies", *Proc. 10th Ann. Int'l Conf. Research in Computational Molecular Biology*, pp. 69-82, 2006.

[8] P.M. Magwene, P. Lizardi, and J. Kim, "Reconstructing the Temporal Ordering of Biological Samples Using Microarray Data", *Bioinformatics*, vol. 19, no. 7, pp. 842-850, 2003.

[9] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, and M.B. Eisen, "Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes", *Molecular Biology of the Cell*, vol. 11, no. 12, pp. 4241-4257, 2000.

[10] D.N. Baldwin, V. Vanchinathan, P.O. Brown, and J.A. Theriot, "A Gene-Expression Program Reflecting the Innate Immune Response of Cultured Intestinal Epithelial Cells to Infection by Listeria Monocytogenes", *Genome Biology*, vol. 4, no. 1, 2003.

[11] C.L. Nutt et al., "Gene Expression-Based Classification of Malignant Gliomas Correlates Better with survival than Histological Classification", *Cancer Research*, vol. 63, no. 7, pp. 1602-1607, 2003.

[12] T.R. Golub et al., "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring", *Science*, vol. 286, pp. 531-537, 1999.

[13] M.J. Bissell et al., "Tissue Structure, Nuclear Organization and Gene Expression in Normal and Malignant Breast", *Cancer Research*, vol. 59, pp. 1757s-1764s, 1999.

[14] C. Jin et al., "Irectionally Specific Paracrine Communication Mediated by Epithelial fgf9 to Stromal fgfr3 in Two-Compartment Premalignant Prostate Tumors", *Cancer Research*, vol. 64, pp. 4555- 4562, 2004.

[15] http://www.ncbi.nlm.nih.gov/sites/entrez?db=omim

[16] http://www.dia.unisa.it/~ads/BIOINFORMATICA/HiddenMarkovModels/Arg4.htm

[17] http://www.dis.uniroma1.it/~rinaldi/teaching/Parte1.pdf

[18] http://en.wikipedia.org/wiki/Line_search

[19] http://www.dii.unisi.it/~agnetis/ottnonvinc.pdf

[20] http://www.dmsa.unipd.it/ ~zilli/Line_search.pdf

[21] Barbara Di Camillo, Brian A. Irving, Jill Schimke, Tiziana Sanavia, Gianna Toffolo, Claudio Cobelli, K.S.Nair. "Function-based discovery of significant transcriptional temporal patterns in insulin stimulated muscle cells". Under revision. 2001.

[22] Alessandra Trojani et al. "ARSD expression and sphingolipid metabolism correlate with IGVH mutational status and ZAP-70 expression in CLL patients". Submitted. 2011.

[23] Hamblin AD, Hamblin TJ (2006). "Functional and prognostic role of ZAP-70 in chronic lymphocytic leukaemia". *Expert Opin. Ther. Targets* **9** (6): 1165–78. doi:10.1517/14728222.9.6.1165. PMID 16300468.

[24] Barbara Di Camillo, Brian A. Irving, Jill Schimke, Tiziana Sanavia, Gianna Toffolo, Claudio Cobelli, K.S.Nair. "Function-based discovery of significant transcriptional temporal patterns in insulin stimulated muscle cells". Under revision. 2001.

[25] Irizarry RA, Hobbs B, Collin F, et al. "Exploration, normalization, and summaries of high density oligonucleotide array probe level data". *Biostatistics.* 2003;4(2):249–264

[26] Di Camillo B, Toffolo G, Nair SK, Greenlund LJ, Cobelli C, "Significance analysis of microarray transcript levels in time series experiments". BMC Bioinformatics 8: S10. 2007.

[27] http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm