Università degli Studi Roma Tre
Facoltà di Scienze M.F.N.
Corso di Laurea in Matematica

Sintesi della Tesi di Laurea in Matematica

Problemi di ottimizzazione combinatoria ed algoritmi per il physical mapping del DNA

Candidata
Irene Olivieri

Relatore
Prof. Marco Liverani

Anno Accademico 2004-2005
Maggio 2006

Classificazione AMS: 68R05, 68R10, 68Q25, 92B05, 92D.

Parole chiave: DNA, Grafi, Algoritmi, Physical Mapping.

Premessa

Il DNA (*acido deossiribonucleico*) è una macromolecola organica presente in tutti gli organismi viventi. La struttura del DNA è, generalmente, una doppia catena di molecole più semplici, dette *nucleotidi*, composte da una molecola di deossiribosio (zucchero semplice), un gruppo fosfato e una base azotata: Adenina (A), Guanina (G), Citosina (C), Timina (T). La singola stringa di DNA è formata da legami fra il deossiribosio e il gruppo fosfato; le due stringhe sono legate fra loro attraverso le basi azotate, con un vincolo, imposto dai possibili legami idrogeno che esse possono formare, tale che si leghino la base A con la T e la C con la G.

Il DNA si presenta nelle cellule in lunghe molecole, i *cromosomi*; tratti contigui di DNA, che contengono le informazioni necessarie per costruire le proteine e le molecole di RNA, sono chiamati *geni*. L'insieme completo dei cromosomi di una cellula è detto *genoma*; il numero di cromosomi in un genoma è una delle caratteristiche di una specie.

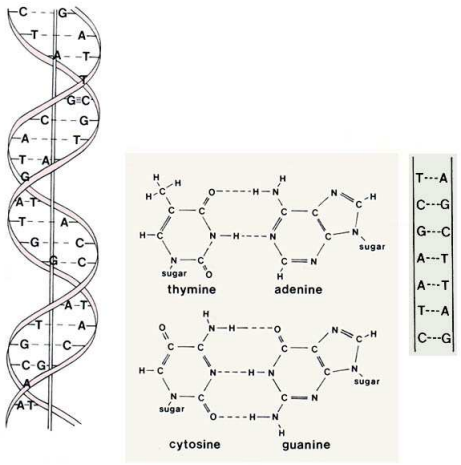

Figura 1: Doppia elica costituita dalle coppie di basi del DNA

Physical mapping

Un importante strumento per lo studio del genoma è il *physical mapping*. Una *physical map* di una molecola di DNA è una 'mappa' che individua la posizione di sequenze note di DNA all'interno della molecola in esame. Si pensi, ad esempio, ad una *physical map* di un cromosoma che ricostruisca la posizione dei geni all'interno di esso. Una delle principali ragioni che ha condotto allo studio del *physical mapping* del DNA è l'impossibilità di ricostruire direttamente la sequenza di coppie di basi componenti frammenti di DNA di lunghezza significativa: dopo aver frammentato la molecola in segmenti, dei quali si possa ricostruire la sequenza di coppie di basi, tramite una *physical map* della molecola di DNA in esame è possibile individuare la posizione dei frammenti all'interno della molecola stessa.

Le principali tecniche utilizzate per ottenere una *physical map* di una molecola di DNA sono l'analisi dei siti di restrizione e l'ibridazione. La mappa dei siti di restrizione individua alcune particolari sequenze (siti di restrizione) in corrispondenza delle quali il DNA viene frammentato da specifici

enzimi, detti enzimi di restrizione. La molecola viene lasciata sotto l'azione di uno (*partial digest problem*) o due enzimi con siti di restrizione differenti (*double digest problem*): i frammenti ottenuti vengono confrontati tra loro, utilizzando come informazione caratterizzante la lunghezza.

La mappa di ibridazione individua l'ordine reciproco dei frammenti di una molecola di DNA, studiandone le sovrapposizioni. La molecola in esame (detta *target DNA*) viene replicata più volte ed ogni copia viene frammentata in numerosi segmenti (detti *cloni*). Mediante la tecnica di ibridazione si verifica se alcune brevi sequenze note, dette *prove*, si leghino (o *ibridino*) a qualche clone; nell'evenienza che ciò accada, si può affermare che il clone contenga la sequenza complementare della sequenza prova. L'informazione che caratterizza un clone è costituita dall'insieme delle prove che ibridano tale frammento. Se, inoltre, due cloni hanno in comune alcune prove, con ogni probabilità essi provengono da zone di sovrapposizione del target DNA. Per mezzo di tali informazioni è possibile ricostruire l'ordine reciproco dei cloni lungo il target DNA.

Nel corso del lavoro che si intende presentare si è scelto di focalizzare l'attenzione sulla seconda delle tecniche citate - la mappa di ibridazione - al fine di descrivere i principali problemi ad essa connessi: il *consecutive ones problem* e due problemi legati ai grafi intervallo (l'*interval sandwich graph problem* e l'*interval coloring graph problem*).

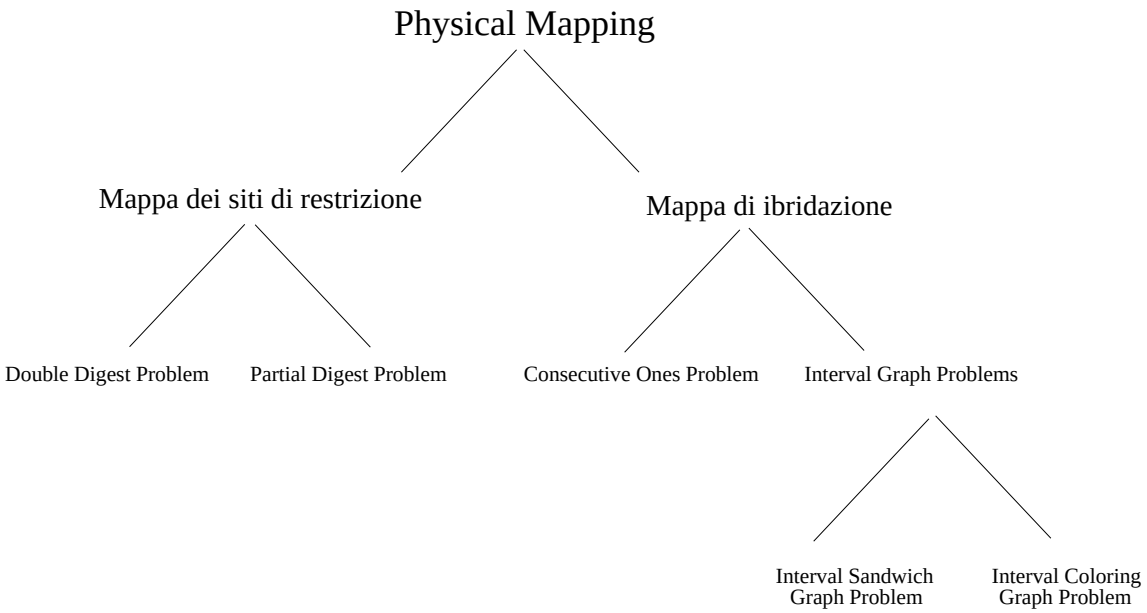

Figura 2: Principali problemi legati al *physical mapping* del DNA

Consecutive ones problem

Si supponga di poter disporre dei risultati di un esperimento condotto su n cloni con m sequenze di prova. Essi possono essere rappresentati tramite una matrice binaria $A \in M_{(n,m)}(\{0,1\})$, in cui le righe rappresentino i cloni e le colonne le prove, tale che l'elemento $A_{ij} = 1$, se la prova j ibrida il clone i ed $A_{ij} = 0$ altrimenti. La costruzione di una *physical map* da tale matrice implica il problema di trovare una permutazione delle colonne (prove), tale che tutti gli elementi pari ad 1 in ogni riga (clone) occupino posizioni consecutive. Se l'esperimento è privo di errori e le prove sono *uniche*[1], la matrice binaria A possiede la *consecutive ones property* (C1P): essa, cioè, ammette una permutazione delle colonne che renda consecutivi, sulle righe, tutti gli elementi pari ad 1. Il modello descritto genera un problema risolvibile in

[1]Una prova è *unica* se si lega ad un solo sito nella molecola in esame.

tempo polinomiale: nel corso della trattazione che in tale sede si intende presentare, è illustrato ed integrato un algoritmo per la risoluzione del suddetto problema; l'algoritmo è stato implementato in linguaggio C. Si è dimostrato, inoltre, che la complessità computazionale di tale programma è dell'ordine di $O(n^3m + n^2m^2)$, con n il numero delle righe della matrice cloni per prove A e m il numero delle colonne.

Algoritmo C1P

L'Algoritmo C1P, data una matrice binaria $A \in M_{(n,m)}(\{0,1\})$, stabilisce se tale matrice soddisfa la *consecutive ones property* e, in caso positivo, determina una permutazione C1P per le colonne di A.

Algoritmo 1 Consecutive ones problem

Input: Matrice $A(n \times m)$
Output: Matrice $C(n \times m)$ ottenuta da A, che verifica la C1P
1: costruisci il grafo delle righe G_A
2: costruisci il grafo delle componenti G_c
3: determina un ordinamento topologico delle componenti
4: **per** ogni componente α (seguendo l'ordinamento topologico) **ripeti**
5: determina una permutazione C1P per la componente α
6: unisci le componenti
7: **fine-ciclo**
8: **se** la matrice A soddisfa la C1P **allora**
9: stampa la matrice permutata C
10: **altrimenti**
11: scrivi: 'la matrice inserita non soddisfa la C1P'
12: **fine-condizione**

Di seguito si illustrerà, tramite un esempio, il funzionamento dell'Algoritmo C1P. Verrano, inoltre, richiamati i principali risultati teorici che garantiscono la correttezza dell'Algoritmo.

Sia $A \in M_{(8,9)}(\{0,1\})$ la matrice cloni per prove:

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Analizzando le relazioni possibili tra le righe della matrice A, $\forall A_i \in A$ (A_i riga i-esima), si definisca l'insieme $S_i := \{A^k \in A : A_{i,k} = 1\}$, con A^k colonna k-esima.

Date due righe $A_i, A_j \in A$, possono verificarsi le seguenti condizioni:

(i) $S_i \cap S_j = \emptyset$

(ii) $S_i \subseteq S_j \vee S_j \subseteq S_i$

(iii) $S_i \cap S_j \neq \emptyset \wedge S_i \nsubseteq S_j \wedge S_j \nsubseteq S_i$

Permutare le colonne di due righe A_i e A_j per le quali valga la condizione (i), in modo tale che valga la la C1P sulla riga A_i, non ha alcun impatto sulla riga A_j, e viceversa. Le due righe, dunque, possono essere elaborate indipendentemente l'una dall'altra.

Se due righe A_i e A_j sono tali che valga la condizione (ii), ad esempio con $S_i \subseteq S_j$, allora le permutazioni delle colonne tali da attribuire la C1P alla riga A_i non hanno alcun impatto sulla riga A_j.

Nel caso in cui tra le righe A_i e A_j sussista la condizione (iii), infine, è necessario porre una certa attenzione nel permutare le colonne della matrice: una permutazione utile ad attribuire la C1P alla riga A_i potrebbe corrompere la sequenzialità degli elementi di valore 1 sulla riga A_j.

Il primo passo dell'algoritmo, dunque, consiste nel caratterizzare tali legami tra le righe della matrice, costruendo un grafo ed individuando le componenti connesse di tale grafo.

Sia $G_A = (V_A, E_A)$ il grafo, *non orientato*, definito dalla matrice A nel modo seguente: $V_A := \{A_i, i = 1, \ldots, n\}$, $(A_i, A_j) \in E_A \Leftrightarrow S_i, S_j$ soddisfano la condizione (iii).

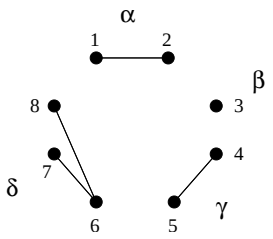

Figura 3: Il grafo delle righe G_A

Come osservato in precedenza, se due righe A_i e A_j sono caratterizzate dalla condizione (ii), ad esempio perché $S_i \subseteq S_j$, eseguire la permutazione delle colonne della matrice A, al fine di rendere sequenziali gli elementi di valore 1 nella riga A_i, successivamente all'imposizione della C1P alla riga A_j, non corrompe tale proprietà su A_j. Non è, tuttavia, vero il contrario: è, dunque, necessario rispettare tale ordine durante l'elaborazione delle permutazioni delle colonne della matrice A, producendo una permutazione compatibile con la C1P per S_j prima che per S_i.

Al fine di rispettare l'ordine suddetto, al secondo passo l'algoritmo costruisce un grafo orientato delle componenti connesse di G_A, i cui spigoli indichino l'ordine con il quale devono essere elaborate le componenti di G_A. Le componenti connesse, infatti, aggregano vertici corrispondenti a righe le cui permutazioni possono interferire vicendevolmente; due righe appartenenti a componenti connesse differenti, invece, possono soddisfare le condizioni (i) e (ii), ma non la (iii).

Sia $G_c = (V_c, E_c)$ il grafo *orientato*, ottenuto dal grafo G_A nel modo

seguente: $V_c := \{\text{componenti connesse di } G_A\}$, $\forall \alpha, \beta \in V_c$, $(\alpha, \beta) \in E_c \Leftrightarrow$ $\forall A_i \in \beta\ \exists A_j \in \alpha$ tale che $S_i \subseteq S_j$.

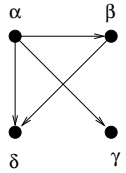

Figura 4: Il grafo delle componenti connesse G_c

Lemma 1 *Il grafo G_c è aciclico*[2]*.*

Si determina, dunque, un ordinamento topologico dei vertici di G_c. L'esistenza di un tale ordinamento è garantita dall'aciclicità del grafo orientato G_c[3].

L'ordinamento topologico dei vertici del grafo G_c considerato dall'esempio rappresentato in figura 4 è: $\alpha < \beta < \gamma < \delta$.

Proposizione 1 *Siano $\alpha, \beta \in V_c$ due componenti connesse distinte del grafo G_A. Sia $\alpha < \beta$ nell'ordinamento topologico dei vertici del grafo G_c. Allora $(\alpha, \beta) \in E_c$, oppure $\forall A_i \in \beta, \forall A_j \in \alpha \Rightarrow S_i \cap S_j = \emptyset$*[4]*.*

La proposizione precedente garantisce che le informazioni ricavabili da una componente connessa β del grafo G_A, riguardanti gli insiemi S_i associati agli elementi A_i che costituiscono tale componente, siano compatibili con le

[2]Per la dimostrazione si veda il Lemma 1 a pagina 18 della Tesi di Laurea.

[3]Gli spigoli di un grafo orientato aciclico G inducono un ordinamento parziale sui vertici dello stesso: $u < v \Leftrightarrow (u, v) \in E_G$. Un *ordinamento topologico* dei vertici di un grafo orientato aciclico G è un ordinamento totale dei vertici compatibile con l'ordinamento parziale indotto dagli spigoli di G. L'ordinamento topologico degli spigoli di un grafo orientato aciclico non è necessariamente unico.

[4]Per la dimostrazione si veda la Proposizione 2 a pagina 18 della Tesi di Laurea.

informazioni ricavabili da una componente connessa minore di β nell'ordinamento topologico dei vertici del grafo G_c. Il probleme viene dunque suddiviso in sottoproblemi che possono essere trattati singolarmente.

Si determina, quindi, una permutazione delle colonne che renda consecutivi gli elementi pari ad 1 sulle righe di ogni singola componente connessa, esaminando le componenti secondo l'ordinamento topologico stabilito. La permutazione per una singola componente connessa α è stata deteminata scorrendo le righe della componente, tramite una visita in profondità, confrontando tra loro le cardinalità degli insiemi S_i associati alle righe i della componente stessa[5].

Nell'esempio considerato le componenti correttamente permutate sono:

componente α:

	{1}	{2}	{4}	{5}	{7}	{9}	{3}	{6}	{8}
$r_1 \rightarrow$	1	1	1	1	1	1	0	0	0
$r_2 \rightarrow$	0	1	1	1	1	1	1	1	1

componente β:

	{2}	{4}	{5}	{7}	{9}	{.}	{.}	{.}	{.}
$r_3 \rightarrow$	1	1	1	1	1	0	0	0	0

componente γ:

	{6}	{3}	{8}	{.}	{.}	{.}	{.}	{.}	{.}
$r_4 \rightarrow$	0	1	1	0	0	0	0	0	0
$r_5 \rightarrow$	1	1	0	0	0	0	0	0	0

componente δ:

	{9}	{5}	{4}	{7}	{2}	{.}	{.}	{.}	{.}
$r_6 \rightarrow$	0	0	1	1	0	0	0	0	0
$r_7 \rightarrow$	0	0	0	1	1	0	0	0	0
$r_8 \rightarrow$	1	1	1	0	0	0	0	0	0

[5]Si rimanda, per una trattazione completa sull'argomento, al paragrafo 2.2.4 e all'Appendice B della Tesi di Laurea

Le componenti, una volta permutate, vengono inserite in una matrice $C \in M_{(n,m)}(\{0,1\})$, che, terminato l'algoritmo, sarà la matrice cloni per prove A, permutata secondo la permutazione C1P. Sia β la componente in esame: nel caso in cui β non sia la prima componente esaminata, è necessario individuare le colonne della matrice C sotto le quali posizionare le colonne della componente contenenti elementi pari ad 1. La permutazione ottenuta per una componente, come osservato precedentemente, è compatibile con le permutazioni ottenute per le componenti che la precedono nell'ordinamento topologico; le colonne di C, le cui etichette corrispondono alle etichette delle colonne della componente contenenti elementi pari ad 1, inoltre, sono tra loro uguali, come garantito dalla proposizione seguente:

Proposizione 2 *Siano $\alpha, \beta \in V_c$ due componenti connesse distinte di G_A. Siano $A_i, A_k \in \beta$, $A_j \in \alpha$. Allora $S_k \subseteq S_j \Leftrightarrow S_i \subseteq S_j$*[6].

Le colonne della componente permutata, contenenti elementi pari ad 1, vengono, dunque, copiate in C a partire dalla colonna più a sinistra tra quelle le cui etichette corrispondano alle colonne della componente contenenti elementi pari ad 1 e le etichette della permutazione della componente vengono sostituite alle etichette delle colonne della matrice C. Esempio:

matrice C

etichette:	{1}	{2}	{4}	{5}	{7}	{9}	{3}	{6}	{8}
α:									
r_1	1	1	1	1	1	1	0	0	0
r_2	0	1	1	1	1	1	1	1	1

[6]Per la dimostrazione si veda la Proposizione 1 a pagina 17 della Tesi di Laurea.

matrice C

etichette:	{1}	{2}	{4}	{5}	{7}	{9}	{3}	{6}	{8}
α:									
r_1	1	1	1	1	1	1	0	0	0
r_2	0	1	1	1	1	1	1	1	1
β:									
r_3	0	1	1	1	1	1	0	0	0

matrice C

etichette:	{1}	{2}	{4}	{5}	{7}	{9}	{6}	{3}	{8}
α:									
r_1	1	1	1	1	1	1	0	0	0
r_2	0	1	1	1	1	1	1	1	1
β:									
r_3	0	1	1	1	1	1	0	0	0
γ:									
r_4	0	0	0	0	0	0	0	1	1
r_5	0	0	0	0	0	0	1	1	0

matrice C

etichette:	{1}	{9}	{5}	{4}	{7}	{2}	{6}	{3}	{8}
α:									
r_1	1	1	1	1	1	1	0	0	0
r_2	0	1	1	1	1	1	1	1	1
β:									
r_3	0	1	1	1	1	1	0	0	0
γ:									
r_4	0	0	0	0	0	0	0	1	1
r_5	0	0	0	0	0	0	1	1	0
δ:									
r_6	0	0	0	1	1	0	0	0	0
r_7	0	0	0	0	1	1	0	0	0
r_8	0	1	1	1	0	0	0	0	0

Interval graph problems

Un grafo non orientato $G = (V, E)$ è un *grafo intervallo* se è possibile definire una corrispondenza biunivoca tra V ed un insieme $\mathcal{I}$ di intervalli su di un insieme dotato di un ordinamento lineare[7], tale che due intervalli si intersechino se e soltanto se i corrispondenti vertici sono adiacenti in G. L'insieme $\mathcal{I}$ è detto *rappresentazione in intervalli* di G.

La costruzione di una *physical map* tramite la tecnica dell'ibridazione conduce ad interessanti problemi relativi ai grafi intervallo: è, infatti, possibile, a partire dai risultati di un esperimento di ibridazione condotto su cloni di una molecola di DNA che evidenzi le sovrapposizioni tra i frammenti, costruire un grafo $G = (V, E)$. Se le informazioni sperimentali sono complete e prive di errori, il grafo $G = (V, E)$, tale che l'insieme dei vertici V rappresenti l'insieme dei cloni e che uno spigolo $(u, v) \in E$ se e soltanto se i cloni rappresentati da u e v si sovrappongono, è un grafo intervallo.

I dati sperimentali, però, contengono generalmente numerosi errori: è possibile, infatti, che una prova si leghi ad un frammento al quale non si sarebbe dovuta legare, creando un *falso positivo*; al contrario è possibile che essa non si leghi ad un clone al quale si sarebbe dovuta legare, creando un *falso negativo*. È inoltre possibile che, durante il processo di clonazione, due frammenti si uniscano, così da venire replicati come un singolo clone (*clone chimerico*). I risultati di un esperimento, dunque, possono risultare viziati, rendendo le le informazioni di sovrapposizione non completamente certe; le coppie di cloni possono essere suddivise in tre gruppi:

i) coppie di cloni la cui sovrapposizione è certa;

ii) coppie di cloni la cui sovrapposizione non è certa;

iii) coppie di cloni che certamente non si sovrappongono.

[7] Un esempio immmediato di un siffatto insieme è la retta reale $\mathbb{R}$.

Tale suddivisione suggerisce la formulazione di un problema appartenente ad una classe di problemi frequentemente utilizzata nella teoria dei grafi: i *sandwich graph problems*.

Il *sandwich graph problem* per la proprietà Π può essere formulato come segue: dati due grafi $G = (V, E)$ e $G_F = (V, F)$ tali che $E \cap F = \emptyset$, esiste un grafo $\bar{G} = (V, \bar{E})$ tale che $E \subseteq \bar{E}$, $\bar{E} \cap F = \emptyset$ e che soddisfi la proprietà Π? Un'istanza di tale problema può essere espressa tramite la tripla $S = (V, E, F)$.

Un caso particolare del *sandwich graph problem* è rappresentato dal *coloring graph problem*: dato un grafo $G = (V, E)$ e una buona colorazione[8] $c : V \longrightarrow \mathbb{N}$ di G, G è un sottografo di un grafo $\bar{G} = (V, \bar{E})$ per il quale c è una buona colorazione e tale che soddisfi la proprietà Π? Il *coloring graph problem* è una specifica istanza di un *sandwich graph problem*, nella quale $F := \{(u, v) \mid c(u) = c(v)\}$.

Siano, dunque, $G = (V, E)$ e $G_F = (V, F)$ i grafi costruiti in modo tale che l'insieme V rappresenti l'insieme dei cloni, E l'insieme delle coppie di vertici appartenenti ad (i) ed F l'insieme della coppie di vertici appartenenti ad (iii). Ne segue, dunque:

INTERVAL SANDWICH GRAPH PROBLEM

Istanza: $S = (V, E, F)$, con V un insieme di vertici e E, F due insiemi disgiunti di spigoli su V.

Problema: Decidere se esiste un grafo intervallo $\bar{G} = (V, \bar{E})$ tale che $E \subseteq \bar{E}$ e $\bar{E} \cap F = \emptyset$.

[8]Una *colorazione* di ordine k per il grafo G è un'applicazione $c : V \to \{1, 2, \ldots, k\}$. Una colorazione per il grafo G è una *buona colorazione* se due vertici adiacenti non hanno mai lo stesso colore: $(u, v) \in E \Rightarrow c(u) \neq c(v)$.

Come istanza particolare dell'*interval sandwich graph problem*, nella quale $F := \{(u, v) \mid c(u) = c(v)\}$, si studia il problema:

INTERVAL COLORING GRAPH PROBLEM

Istanza: Un grafo $G = (V, E)$ e una buona colorazione $c : V \longrightarrow \mathbb{N}$ di G.

Problema: Decidere se G è un sottografo di un grafo intervallo $\bar{G} = (V, \bar{E})$ per il quale c è una buona colorazione.

Algoritmo SANDWICH

In questo paragafo verrà illustrato un algoritmo per l'*interval sandwich graph problem*. Prima di introdurre l'algoritmo verranno fornite alcune definizioni e i principali risultati, necessari al fine di comprendere l'operato di tale algoritmo.

Sia $S = (V, E, F)$ un'istanza del problema sandwich. Un *taglio* di S è un sottoinsieme $X \subseteq V$. L'insieme degli spigoli $\delta(X)$ di un taglio X è costituito dagli spigoli $(u, v) \in E$ tali che $u \in X$ e $v \in V \setminus X$. L'*active region* di X, $\Delta(X)$, è un sottoinsieme di X costituito dai vertici incidenti ad almeno uno spigolo appartenente a $\delta(X)$. Un *layout* del taglio X è una funzione che associa ad ogni vertice $v \in X$ un intervallo $I(v)$ in modo tale che valgano le seguenti condizioni:

1. $I(v) \cap I(u) \neq \emptyset$, se $(v, u) \in E$;
2. $I(v) \cap I(u) = \emptyset$, se $(v, u) \in F$;
3. $r(I(v)) = max\{r(I(w)) \mid w \in X\}, \forall\ v \in \Delta(X)$, con $r(I(u))$ estremo destro dell'intervallo assegnato ad u.

Un taglio di S è *realizzabile* se ammette un layout. Un taglio realizzabile Y di S *estende* un taglio realizzabile $X \neq Y$ se $Y = X \cup \{v\}$ ed è possibile ottenere un layout per Y da un layout per X aggiungendo l'intervallo $I(v)$ alla destra di tutti gli intervalli appartenenti a $L(X)$, cambiando in $r(I(v))$ gli estremi destri di tutti gli intervalli associati ai vertici in $\Delta(X)$.

Il lemma seguente fornisce una caratterizzazione della relazione di estensione tra tagli:

Lemma 2 *Sia X un taglio realizzabile e $Y = X \cup \{v\}$. Y è un taglio realizzabile che estende X se e soltanto se $\forall\ w \in \Delta(X)$ si ha che $(v, w) \notin F$.*

I lemmi seguenti mostrano che, tramite estensioni successive di un taglio realizzabile, è possibile stabilire se esiste una soluzione dell'istanza del problema:

Lemma 3 *Ogni taglio realizzabile estende almeno un taglio realizzabile di cardinalità minore.*

Lemma 4 *Una soluzione dell'istanza $S = (V, E, F)$ dell'interval sandwich problem esiste se e soltanto se V è un taglio realizzabile.*

Alla luce delle precedenti considerazioni, un algoritmo che trovi, se esiste, una soluzione per un'istanza $S = (V, E, F)$ dell'*interval sandwich problem* cercherà, tramite estensioni successive, un taglio realizzabile che coincida con l'insieme V.

Algoritmo 2 SANDWICH (S)

```
1: sia S = (V, E, F) un'istanza dell'interval sandwich problem
2: si inizializzi una coda Q = ∅
3: per ogni vertice v ∈ V ripeti
4:    si aggiunga il taglio realizzabile {v} alla coda Q
5: fine-ciclo
6: fintanto che Q ≠ ∅ ripeti
7:    si rimuova un taglio realizzabile X dalla coda
8:    per ogni vertice z ∉ X ripeti
9:       si provi ad estendere il taglio X tramite z (utilizzando il Lemma 2)
10:      se Y = X ∪ {z} è un taglio realizzabile allora
11:         se Y = V allora
12:            si scriva in output 'successo' e STOP
13:         altrimenti se Y non si trova già in Q allora
14:            aggiungi Y a Q
15:         fine-condizione
16:      fine-condizione
17:   fine-ciclo
18: fine-ciclo
```

L'Algoritmo SANDWICH genera tutti i tagli realizzabili in ordine di cardinalità crescente: mantiene una coda Q di tagli realizzabili che possano essere ancora estesi. In ogni iterazione elimina dalla coda un taglio X e genera tutti i tagli realizzabili che estendono X; ogni estensione che viene trovata per la prima volta viene inserita in Q. Una soluzione per l'istanza data esiste se e soltanto se il taglio V viene generato prima che la coda Q si sia svuotata.

Si assuma che l'algoritmo mantenga una tabella nella quale memorizza ogni taglio già esaminato, affinchè si possa stabilire in tempo costante se il taglio generato è già in Q o meno. Si assuma inoltre che per ogni coppia di vertici l'esistenza di uno spigolo proibito fra essi possa essere rilevata in tempo costante.

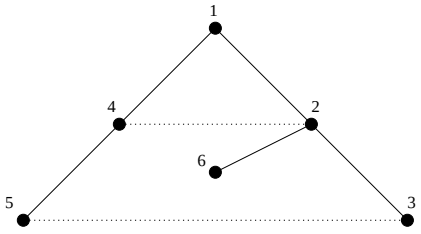

Figura 5: Istanza $S = (V, E, F)$ dell'interval sandwich problem

Sia G il grafo in figura 5: gli spigoli disegnati in linea continua sono gli spigoli appartenenti all'insieme E, mentre gli spigoli disegnati in linea tratteggiata appartengono all'insieme F. L'algoritmo SANDWICH, dopo aver inizializzato la coda Q (passi $3-5$) con i singleton $\{v\}, \forall v \in V$, genera tutti i tagli realizzabili in ordine di cardinalità crescente.

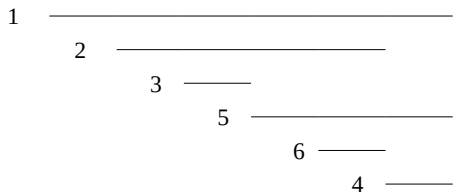

Figura 6: Layout $L(V)$

Il layout $L(V)$, mostrato in figura 6, è il layout costruito, tramite estensioni successive, a partire dal taglio realizzabile $\{1\}$. I vertici sono mostrati in ordine di estensione. Il grafo intervallo soluzione dell'istanza considerata, una rappresentazione in intervalli del quale è costituita dal layout $L(V)$, è mostrato in figura 7.

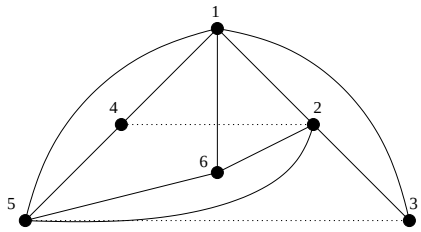

Figura 7: Grafo intervallo soluzione dell'istanza $S = (V, E, F)$

Il tempo di esecuzione dell'algoritmo descritto è proporzionale al numero dei tagli realizzabili che, nel caso peggiore, è esponenziale. L'*interval sandwich graph problem* e l'*interval coloring graph problem*, infatti, appartengono alla classe dei problemi NP-completi; porre delle opportune restrizioni a tali problemi, tuttavia, li rende risolvibili in tempo polinomiale. Le restrizioni considerate scaturiscono dall'osservazione di favorevoli caratteristiche dei reali dati biologici: la 'profondità' delle mappe è costantemente molto piccola, ossia il maggior numero di cloni che mutuamente si sovrappongono è tipicamente compreso tra 5 e 15, mentre i cloni complessivamente utilizzati in un esperimento sono nell'ordine delle migliaia.

Le considerazioni precedenti implicano che i grafi costruiti tramite i dati sperimentali siano molto sparsi: in particolare, la 'profondità' massima di una mappa corrisponde alla dimensione della massima clique[9] del grafo modello.

Nel corso della trattazione che in tale sede si intende presentare, sono state discusse due parametrizzazioni per l'*interval sandwich graph problem* (e le relative parametrizzazioni per l'*interval coloring graph problem*); per tali parametrizzazioni, sono stati, inoltre, illustrati due algoritmi polinomiali, ogni qual volta i parametri siano fissati.

Parametric interval sandwich graph problem con clique e grado limitati:
Data un'istanza $S = (V, E, F)$, con V un insieme di vertici, E ed F due insiemi disgiunti di spigoli su V, tali che il grado di ogni vertice sia limitato da una costante d, determinare se esiste un grafo intervallo $\bar{G} = (V, \bar{E})$ tale che $E \subseteq \bar{E}$, $\bar{E} \cap F = \emptyset$ e che la dimensione della clique massima sia limitata da una costante k.

In tale modello vengono limitati la dimensione della clique massima del grafo sandwich e il grado massimo del grafo di input. Limitare la dimensione della clique massima nel grafo sandwich significa imporre un limite al massimo numero di cloni che si sovrappongono mutuamente nella mappa soluzione del problema; limitare il grado massimo dei vertici del grafo di input significa chiedere che nei dati sia limitato il numero massimo dei cloni che certamente intersecano lo stesso clone.

L'algoritmo presentato per tale modello ha una complessità computazionale dell'ordine di $O(n^{k-1})$, per d e k fissati.

[9]Una *clique* in un grafo $G = (V, E)$ è un sottoinsieme $C \subseteq V$ tale che il sottografo indotto da C sia completo. Una clique è *massimale* se non è contenuta propriamente in nessun'altra clique del grafo. La clique *massima* è la clique (non necessariamente unica) di cardinalità massima.

Parametric interval sandwich graph problem con grado del grafo intervallo limitato:
Data un'istanza $S = (V, E, F)$, con V un insieme di vertici, E ed F due insiemi disgiunti di spigoli su V, determinare se esiste un grafo intervallo $\bar{G} = (V, \bar{E})$ tale che $E \subseteq \bar{E}$, $\bar{E} \cap F = \emptyset$ e che il grado di ogni vertice sia limitato da una costante d.

In tale modello viene limitato il massimo grado dei vertici del grafo sandwich richiesto: ciò equivale a richiedere che nella mappa soluzione del problema sia limitato il numero massimo di cloni che può intersecare lo stesso frammento.

L'algoritmo presentato per tale modello ha una complessità computazionale dell'ordine di $O(n^{d-1})$, per d fissato.

Bibliografia

[1] A. Brandstadt, Van Bang Le, J.P. Spinrad, *Graph classes: a survey*, SIAM Monographs on Discrete Mathematics and Applications, 1999.

[2] P. V. Ceccherini , *Appunti di Istituzioni di Geometria superiore: Teoria dei Grafi*, Dipartimento di Matematica 'Guido Castelnuovo', Università di Roma 'La Sapienza', 1998.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduzione agli algoritmi e strutture dati*, seconda edizione, McGraw-Hill, 2005.

[4] P. C. Gilmore, A. J. Hoffman, *A characterization of comparability graphs and of interval graphs*, Canad. J. Math., **16**, 539-548, 1964.

[5] F. Harary, *Graph Theory*, Addison Wesley, 1969.

[6] M. C. Golumbic, H. Kaplan, R. Shamir, *On the complexity of DNA physical mapping*, Advances in Applied Mathematics, **15**, 251-261, 1994.

[7] J. Setubal, J. Meidanis, *Introduction to computational molecular biology*, PWS Publishing Company, Boston, New York, Paris, 1997.

[8] H. Kaplan, R. Shamir, *Bounded Degree Interval Sandwich Problems*, Algorthmica, **24**, 96-104, 1999.

[9] H. Kaplan, R. Shamir, *Pathwidth, Bandwidth and Completion Problems to Proper Interval Graphs with Small Cliques*, SIAM Journal on Algorithms, **25**, 540-561, 1996.

[10] J. Opatrny, *Total ordering problems*, SIAM Journal of Computing, **8**, 111-114, 1979.

[11] T. J. Schaefer, *The complexity of satisfiability problems*, Proc. 10th Annual ACM Symp. on Theory of Computing, 216-226, 1978.

[12] R. J. Wilson, *Introduction to graph theory*, quarta edizione, Longman, 1996.