Note del Polo
-Progetti-

14

Utilizzo di algoritmi su grafo per l'analisi di reti di legami a idrogeno in sistemi biologici

Andrea Miracca
Marco Rossetti
Diego Vicoletti

Università degli Studi di Milano

DIPARTIMENTO DI TECNOLOGIE DELL'INFORMAZIONE
POLO DIDATTICO E DI RICERCA DI CREMA

INDICE

1. Introduzione

Il lavoro oggetto della presente pubblicazione è il risultato di un progetto di ricerca nell'ambito dei corsi di **Bioinformatica** e **Complementi di Algoritmi** coordinato dai professori **Sandro Fornili** e **Roberto Cordone** che ringraziamo per la gentile disponibilità.

Lo scopo di questo rapporto interno è quello di presentare i risultati ottenuti analizzando il pattern di legami idrogeno (*hydrogen bonds, HBs)* tra molecole di acqua e tra queste e molecole di soluto presenti in soluzioni acquose. Tali legami sono particolarmente importanti perché condizionano la struttura e la dinamica sia del solvente che del soluto. Infatti struttura e dinamica del solvente e del soluto si influenzano a vicenda e hanno rilevanza fondamentale nel funzionamento di biomolecole e in particolare di proteine.

Ogni atomo di idrogeno del sistema legato a un atomo *donore* può stabilire un legame a idrogeno con un atomo *accettore* appartenente alla stessa molecola o ad una molecola differente.

Tipicamente, donori e accettori sono atomi di ossigeno, ma possono essere anche atomi di azoto, e talvolta zolfo. Per convenzione, si ritiene che sia presente un legame a idrogeno quando la distanza fra donore e accettore è inferiore a una data soglia d_m e l'angolo formato dalla terna donore-idrogeno-accettore supera una data soglia α_m (per esempio $d_m = 3.5$ Å $\alpha_m = 150°$ [11]).

La presenza di una estesa rete di HBs caratterizza il funzionamento dell'acqua sia solida (ghiaccio) che liquida. È quindi opportuno studiare come la presenza di soluti influenzi tale rete.

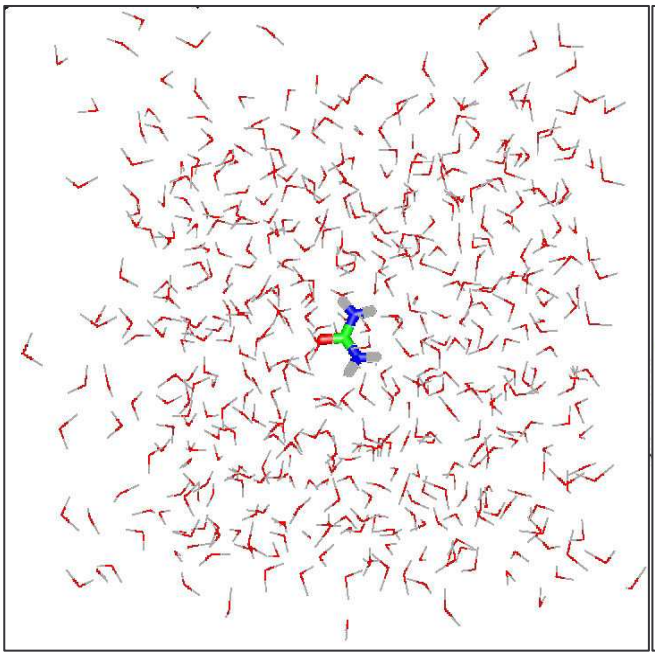


Figura 1.1 - Molecola di urea in soluzione con 500 molecole di acqua

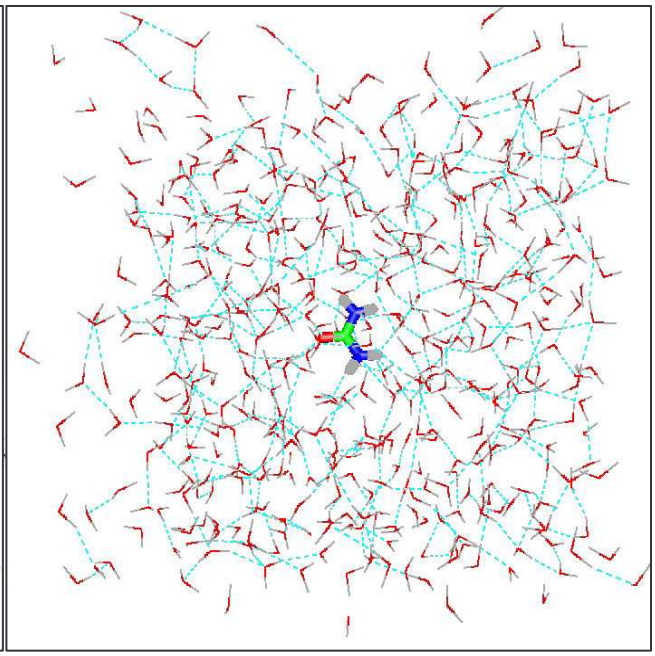


Figura 1.2 – Lo stesso sistema rappresentato in figura 1.1, in cui gli HBs sono rappresentati da segmenti tratteggiati

In Fig. 1.1 viene rappresentata una configurazione di un sistema costituito da 1 molecola di urea e 500 molecole di acqua, mentre in Fig. 1.2 vengono rappresentati gli HBs per lo stesso sistema. Come si può notare, si tratta di un sistema piuttosto complesso che presenta una estesa rete di legami a idrogeno. Questo perché l'acqua esercita un ruolo attivo sulle biomolecole di soluto, che a loro volta influiscono sull'acqua, ed è quindi interessante caratterizzare l'hydrogen bonding dell'acqua intorno alle molecole in soluzione.

I soluti (o parti di essi) possono essere suddivisi in:

- **Idrofilici**: soluti che stabiliscono il contatto con le molecole d'acqua e formano legami a idrogeno (sono elettricamente carichi o polari, p.es. OH).
- **Idrofobici**: soluti che non tendono a formare legami a idrogeno con le molecole d'acqua (sono elettricamente neutri o apolari, come i gruppi CH). In questo caso si formano legami a idrogeno solo tra molecole d'acqua.
- **Anfifilici**: soluti che possono avere sia parti filiche che parti fobiche.

In quest'ultimo caso è interessante vedere come i soluti anfifilici influenzino la distribuzione dei legami a idrogeno nel proprio intorno, in particolare la differenza della quantità di legami a idrogeno tra la parte idrofobica e quella idrofilica. Un esempio semplice ed interessante per capire come si comporta l'acqua attorno ai soluti anfifilici è dato dal metanolo (vedi fig. 1.3).

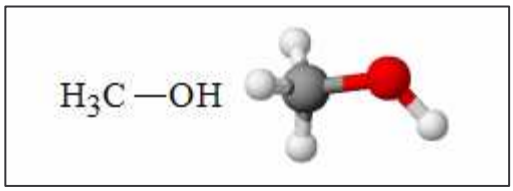


Figura 1.3 - Formula di struttura e modello molecolare del metanolo [1]

Nella fig. 1.4 è rappresentata la distribuzione 3D degli atomi di ossigeno delle molecole di acqua che circondano il metanolo in soluzione acquosa. Appare chiaro come le molecole d'acqua tendano ad addensarsi attorno al gruppo idrofilico (OH) del metanolo, al contrario, il gruppo metilico (CH_3), essendo idrofobico, non interagisce con le molecole d'acqua.

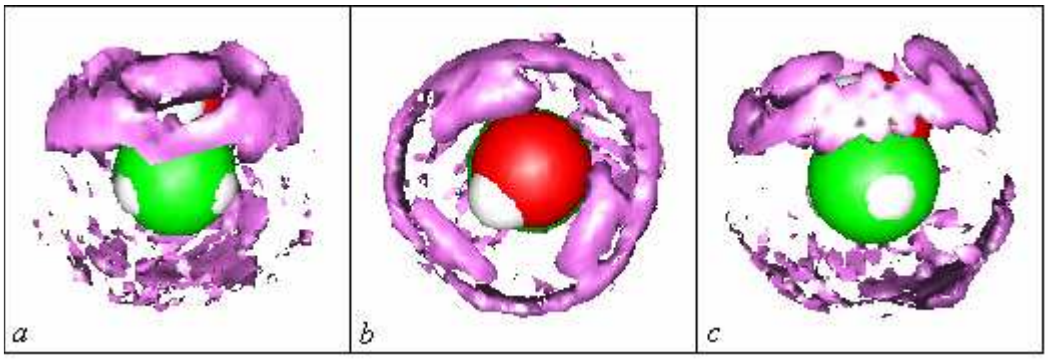


Figura 1.4 - Distribuzione spaziale degli atomi di ossigeno dell'acqua attorno alla molecola di metanolo. g(r) spaziale con livello 1,8 Å [2].

In questo progetto sono stati messi a punto degli algoritmi che permettono di individuare i legami a idrogeno, i *cluster* (cioè gli aggregati di molecole collegate tra loro da HBs) e di studiare le statistiche sui cluster e il decadimento dei legami in funzione del tempo.

I dati analizzati sono stati ottenuti da una serie di *snapshots* derivanti da una simulazione di dinamica molecolare. Questi dati sono contenuti in file in formato PDB. Il PDB (*Protein Data Bank*) è un archivio di dati che descrivono la struttura tridimensionale di oggetti proteici [3].
Il sistema simulato è costituito da una molecola di urea (vedi fig. 1.5) e da 500 molecole di acqua (vedi figg. 1.1, 1.2). Come modello per l'acqua è stato utilizzato il TIP3P [4].

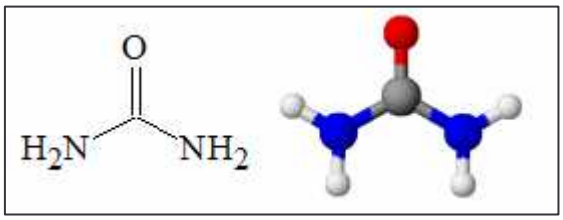


Figura 1.5 - Formula di struttura e modello molecolare dell'urea [5]

L'interesse dello studio di sistemi acquosi di urea deriva dal fatto che, ad alta concentrazione, questa sostanza è un potente denaturante delle proteine (ne rompe cioè la struttura tridimensionale). Il meccanismo di questo effetto non è ancora chiarito, in particolare non si capisce perché questa sostanza che possiede diversi gruppi filici, ad alta concentrazione, sembra presentare caratteristiche tipiche di sostanze fobiche.

In fig. 1.6 è rappresentata la distribuzione 3D degli atomi di ossigeno delle molecole di acqua che circondano l'urea in soluzione acquosa. In fig. 1.6a viene mostrata una visione frontale della molecola, in fig. 1.6b la distribuzione di molecole d'acqua intorno ad un azoto e in fig. 1.6c l'addensamento delle molecole d'acqua sopra l'ossigeno.

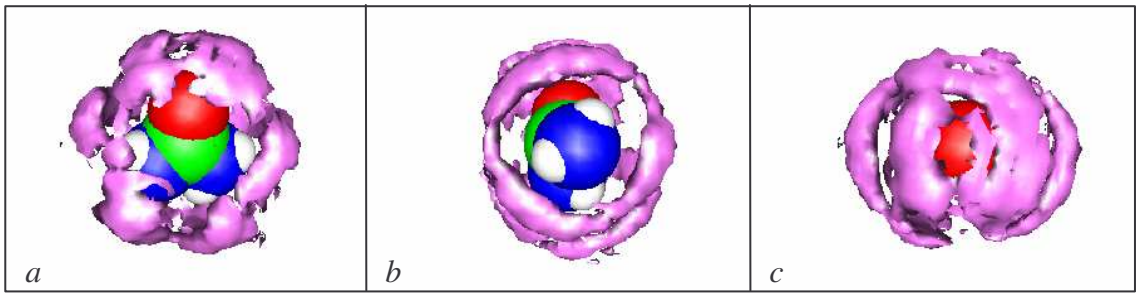


Figura 1.6 - Distribuzione spaziale degli atomi di ossigeno dell'acqua attorno alla molecola di urea. g(r) spaziale con livello 1,8 Å [6].

2. Modellazione

In questo Capitolo verrà illustrato come è stato definito il modello per rappresentare il soluto e il solvente. Inizialmente verrà illustrato il modello dal punto di vista teorico e successivamente ne sarà descritta l'implementazione mediante le strutture dati, infine saranno descritti gli algoritmi utilizzati per l'individuazione dei legami a idrogeno, per l'individuazione dei cluster, per le statistiche sui cluster e per l'analisi del decadimento dei legami.

2.1 Modello teorico

Si è voluta una rappresentazione che rendesse conto della struttura indotta dagli HBs sulla soluzione, in particolare prendendo in considerazione i cluster tra le molecole della soluzione. Come unità elementari sono state scelte le molecole, non gli atomi, in quanto i legami a idrogeno vengono considerati come legami tra molecole. Quindi si è scelto di rappresentare la soluzione con un multigrafo non orientato $G(V,E)$, dove V è l'insieme dei vertici del grafo e E è un insieme di coppie ordinate di vertici dette lati. I vertici, come illustrato in Figura 2.1, rappresentano le molecole che compongono la soluzione, mentre i lati rappresentano i legami a idrogeno. Si tratta di un multigrafo perché possono stabilirsi più legami a idrogeno fra le stesse due molecole, e addirittura fra gli stessi due atomi di molecole diverse, ovvero un singolo atomo (accettore o donore) può dare vita a più legami. Ogni molecola è rappresentata come tre sottoinsiemi di atomi, caratterizzati dal loro ruolo nel legame a idrogeno:

- gli atomi accettori e/o donori;
- gli atomi di idrogeno;
- tutti gli atomi restanti.

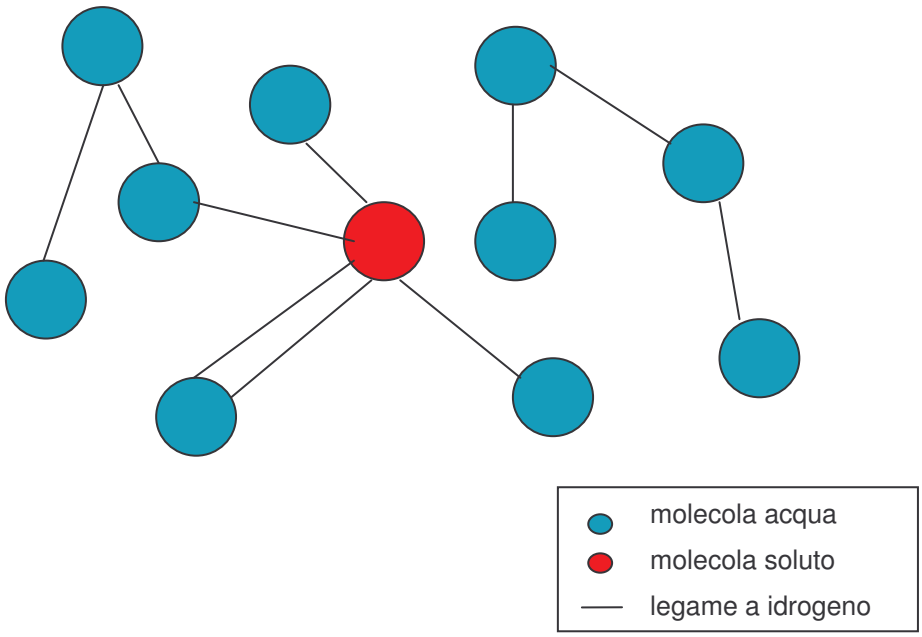


Figura 2.1 - Esempio del grafo

Il grafo, in generale, non è completamente connesso, ma può essere costituito da diverse componenti connesse, ognuna delle quali rappresenta un cluster di molecole.

2.2 Strutture dati

Il grafo descritto in precedenza è stato implementato mediante una struttura dati contenente:

- il numero di molecole;
- il numero di legami a idrogeno;
- il numero di cluster;
- una lista di molecole;
- una lista di cluster.

Ciascuna lista è stata implementata come una doppia lista circolare con sentinella, realizzata attraverso puntatori: ogni elemento punta all'elemento precedente e all'elemento successivo. Quando la lista è vuota contiene solo un elemento fittizio, detto sentinella.

Ogni molecola è implementata come una struttura che contiene:

- un indice, che identifica univocamente la molecola;
- la sostanza chimica;
- una lista degli atomi accettori e/o donori;
- una lista degli atomi di idrogeno;
- una lista degli atomi restanti;
- una lista di legami a idrogeno che interessano la molecola;
- un puntatore al cluster a cui appartiene la molecola.

Infine una coppia di puntatori, rispettivamente alla molecola successiva e alla molecola precedente, che occorrono per scandire la lista delle molecole.

Ogni atomo è rappresentato da una struttura composta dai seguenti campi:

- un indice, che identifica univocamente l'atomo;
- il nome dell'elemento chimico dell'atomo;
- le tre coordinate della posizione dell'atomo nello spazio (in Angstrom);
- un puntatore alla molecola a cui l'atomo appartiene.

Oltre ovviamente alla coppia di puntatori rispettivamente all'atomo successivo e all'atomo precedente che occorrono per scandire la lista degli atomi.

Ogni legame a idrogeno è contenuto nella lista dei legami delle molecole che lo riguardano, ed è descritto dalla seguente struttura:

- un indice, che identifica univocamente il legame;
- la distanza, in angstrom, tra l'atomo accettore e l'atomo donore;
- il coseno dell'angolo tra gli atomi donore-accettore-idrogeno;
- tre puntatori rispettivamente all'atomo accettore, all'atomo donore e all'idrogeno;
- due puntatori alle due molecole coinvolte nel legame;
- un puntatore al legame opposto, ovvero al legame che parte dall'altra molecola coinvolta nel legame.

Infine la coppia di puntatori rispettivamente al legame successivo e al legame precedente che occorrono per scandire la lista dei legami.

Infine è stata utilizzata una struttura per rappresentare ciascun cluster appartenente alla lista dei cluster:

- un indice, che identifica univocamente il cluster;
- il numero delle molecole che compongono il cluster;
- il numero delle molecole d'acqua;
- il numero delle molecole di soluto;
- un puntatore ad una lista d'appoggio, composta da puntatori a molecole;
- una coppia di puntatori rispettivamente al cluster successivo e al cluster precedente contenuti nella lista dei cluster.

2.3 Algoritmi utilizzati

Nelle sezioni seguenti sono descritte le fasi successive del progetto, ovvero gli algoritmi per:

- il caricamento dei dati nelle strutture descritte in precedenza;

- l'individuazione dei legami a idrogeno;
- l'individuazione dei cluster;
- le statistiche sui cluster;
- l'analisi del decadimento dei legami in funzione del tempo (in tutta la soluzione e in prima shell);
- l'analisi della distribuzione delle molecole d'acqua in funzione del numero degli HBs (in tutta la soluzione e in prima shell).

Questi algoritmi sono suddivisi in due parti, corrispondenti a due eseguibili distinti, la prima che comprende i primi tre punti mentre l'ultimo punto è stato sviluppato nella seconda parte.

2.3.1 Caricamento dei dati

L'analisi si basa sui dati contenuti in uno o più file di testo che obbediscono al formato standard PDB. Inizialmente vengono caricati i dati contenuti in un file in formato PDB, che rappresenta un singolo snapshot, all'interno delle strutture dati descritte in precedenza. Questo caricamento è stato effettuato leggendo ogni singola riga del file PDB, che corrisponde ad ogni singolo atomo. Se l'indice della molecola, della riga che si sta leggendo, è presente nella lista delle molecole, allora l'atomo viene aggiunto in coda ad una delle tre liste degli atomi della molecola stessa, in base all'elemento chimico; altrimenti viene aggiunta una nuova molecola alla lista in cui l'atomo della riga letta è inserito in testa alla lista appropriata della nuova molecola.
Per quanto riguarda le statistiche sui cluster e l'analisi del decadimento dei legami in funzione del tempo sono stati realizzati appositi eseguibili che effettuano il caricamento di diversi snapshots da file diversi.

2.3.2. Individuazione dei legami ad idrogeno

Per l'individuazione dei legami a idrogeno si scandiscono tutte le terne di atomi che possono dar luogo a legami, verificando per ciascuna se il legame esiste o no. In particolare è stata fatta la scansione della lista delle molecole. Per ogni molecola (M) è stata scandita la lista degli atomi d'idrogeno, per ogni atomo d'idrogeno (H) viene scandita la lista degli atomi accettori-donori della stessa molecola. Per ogni atomo accettore-donore (D) di quest'ultima lista è stata fatta la scansione della lista delle molecole; quindi per ogni molecola (M1), con M1 diversa da M, è stata fatta una scansione della lista degli atomi accettori-donori. Ogni atomo (A) di quest'ultima lista determina con l'idrogeno H e il donore D una terna potenzialmente associata a un legame. Per determinare l'esistenza del legame a idrogeno sono stati calcolati i seguenti valori:

- distanza tra atomo donore (D) e atomo accettore (A), calcolata sulla base delle coordinate cartesiane, mediante la seguente formula:

$$\|AD\| = \sqrt{(x_D - x_A)^2 + (y_D - y_A)^2 + (z_D - z_A)^2} \qquad (1)$$

- coseno dell'angolo DĤA tra donore-idrogeno-accettore, calcolato sulla base delle coordinate nello spazio degli atomi, mediante la seguente formula:

$$\frac{HD \cdot HA}{\|HD\| \times \|HA\|} = \frac{(x_D - x_H) \times (x_A - x_H) + (y_D - y_H) \times (y_A - y_H) + (z_D - z_H) \times (z_A - z_H)}{\|HD\| \times \|HA\|} \qquad (2)$$

  dove HD è il vettore posizione dell'atomo H rispetto all'atomo D, HA è il vettore posizione dell'attomo H rispetto all'atomo A, e $\|HD\|$ e $\|HA\|$ sono i loro moduli, calcolati tramite l'utilizzo della (1).

In particolare, il legame esisterà se la distanza DA è minore o uguale ad un valore ALFA e se il coseno dell'angolo DĤA è minore o uguale ad un valore DELTA e quindi l'angolo DĤA è maggiore o uguale a un dato angolo limite. ALFA e DELTA sono parametri specificati all'interno di un file di configurazione e in questo caso sono stati assegnati i seguenti valori (tratti dalla letteratura):

- ALFA = 3.5 Å;
- DELTA = -0.866025, corrispondente ad un angolo limite di 150°.

Se queste condizioni sono verificate, allora viene aggiunto il legame M-M1 alla lista dei legami a idrogeno della molecola M; siccome per la rappresentazione è stato utilizzato un grafo non orientato, viene aggiunto anche il legame opposto M1-M.

2.3.3 Individuazione dei cluster

Un cluster è un insieme di molecole legate fra loro, direttamente o indirettamente, da legami a idrogeno. Se si rappresentano le molecole come vertici di un grafo, il concetto di cluster diviene equivalente a quello di *componente connessa* del grafo stesso. Quindi per la loro determinazione si può utilizzare un qualsiasi algoritmo di visita al fine di individuare le componenti connesse. In particolare si è scelto di utilizzare una visita di tipo DFS. Di seguito riportiamo lo pseudocodice della visita DFS di un generico grafo G (V,E):

```
void DFS ( grafo G, nodo u ) {
  nodo v;

  {esame nodo u e marcalo visitato};
  for each v∈V(u) {
     {esamina arco (u,v)};
     if ( v non visitato )
          then DFS (G,v);
  }
}
```

Per l'individuazione dei cluster è stata fatta la scansione della lista delle molecole. Per ogni molecola è stato controllato se essa appartiene già ad un cluster, verificando che il suo puntatore al cluster non sia nullo; in caso negativo, si crea un cluster vuoto il quale viene inserito in coda alla lista dei cluster del grafo. Ovviamente il numero dei cluster viene incrementato di uno, ed infine è chiamata la procedura `Visita` che individua ricorsivamente tutte le altre molecole che appartengono allo stesso cluster della molecola. In questa procedura è posto a `TRUE` il campo "visitato" della molecola corrente, viene creato un puntatore alla molecola corrente che viene inserito in coda alla lista dei puntatori a molecole del cluster, viene incrementato il contatore del numero delle molecole del cluster, quindi la molecola in considerazione viene fatta puntare al cluster. In seguito è controllato se la molecola in esame è di acqua o di soluto, e viene incrementato il relativo contatore.

Viene poi effettuata la scansione della lista dei legami a idrogeno che coinvolgono la molecola corrente. Per ogni molecola, raggiunta da questa scansione, e non ancora visitata per l'individuazione dei cluster, viene chiamata ricorsivamente la procedura `Visita`.

Infine, viene creato il file di output nel formato PDB. Questo file contiene gli stessi atomi e molecole del file originale. Le molecole sono riordinate affinché quelle appartenenti allo stesso cluster siano consecutive. Inoltre, in aggiunta al file originale, sono riportati i cluster individuati attraverso l'utilizzo di etichette `REMARK`. Di seguito è mostrato un esempio di utilizzo di tale etichetta:

```
REMARK 4    CLUSTER 1 MOL_INIZ 1 MOL_FINALE 443
```

La parola chiave `REMARK 4` viene utilizzata per aggiungere annotazioni di testo all'interno di file PDB. Nel seguito della riga sono riportati gli indici per identicare univocamente il cluster, della prima e dell'ultima molecola da cui esso è formato.

2.3.4 Statistiche sui cluster

L'aspetto che più interessa per le statistiche dei cluster riguarda la loro dimensione, cioè il numero di molecole di cui ciascun cluster è formato, e la loro composizione, cioè se siano composti solo da molecole d'acqua, solo da molecole di soluto, oppure sia da molecole d'acqua sia da molecole di soluto. È interessante, in particolare, analizzare come questi aspetti varino nel tempo per una stessa soluzione. I risultati dell'analisi sono riportati in un file di testo (file delle frequenze) strutturato come una tabella. Un esempio di riga della tabella è di seguito rappresentato:

```
26 5 2 2 5   26 5 2 2 4   0 0 0 0 0   0 0 0 0 1
```

Nell'esempio sopra, il primo gruppo di cinque numeri rappresenta la frequenza su tutti i cluster:

in particolare, il primo numero (26) sono i cluster composti da 1 sola molecola, il secondo numero (5) i cluster composti da 2 molecole, il terzo (2) da 3, il quarto (2) da 4, il quinto numero (5) è dato dai cluster aventi 5 o più molecole. Il valore della massima dimensione specificata singolarmente per un cluster (nella riga di esempio tale valore è 5) è una soglia che può essere modificata mediante il parametro SOGLIA nel file di configurazione. Questo valore è di default impostato a 20.
Il secondo gruppo di cinque numeri si riferisce ai cluster con solo molecole di acqua, il terzo ai cluster sia con molecole di soluto che di acqua e, infine, il quarto ai cluster solo con molecole di soluto. In effetti la somma dei numeri della colonna i-esima di questi ultimi tre gruppi dà il valore contenuto nella colonna i-esima del primo gruppo.
Il programma viene eseguito in sequenza su diversi snapshots, cioè su diversi file PDB, per ognuno di questi viene aggiunta (*append*) una riga al file delle frequenze. In questo modo in esso si crea la tabella descritta sopra.
Per implementare quanto spiegato in precedenza sono stati utilizzati quattro array, di dimensione SOGLIA:

- `Freq`, dove l'i-esimo elemento conterrà il numero di cluster composti da i molecole;
- `Wat`, dove l'i-esimo elemento conterrà il numero di cluster composti da i molecole di sola acqua;
- `Sol`, dove l'i-esimo elemento conterrà il numero di cluster composti da i molecole di solo soluto.
- `Ws`, dove l'i-esimo elemento conterrà il numero di cluster composti da i molecole di acqua e di soluto;

Tutti gli elementi degli array sono inizializzati a 0.
Viene effettuata una scansione della lista dei cluster contenuti nel grafo. Per ogni cluster si controlla il numero di molecole, se questo è maggiore di SOGLIA, viene incrementato l'ultimo elemento dell'array `Freq`, altrimenti viene incrementato l'elemento dell'array `Freq` il cui indice corrisponde al numero di molecole.
In seguito si controlla che il cluster contenga solo molecole di acqua. In questo caso si controlla il numero di molecole di acqua, se questo è maggiore di SOGLIA, viene incrementato l'ultimo elemento dell'array `Wat`, altrimenti viene incrementato l'elemento dell'array `Wat` il cui indice corrisponde al numero di molecole di acqua.
Se invece il cluster contiene solo molecole di soluto, si controlla il numero di molecole di soluto, se questo è maggiore di SOGLIA, viene incrementato l'ultimo elemento dell'array `Sol`, altrimenti viene incrementato l'elemento dell'array `Sol` il cui indice corrisponde al numero di molecole di soluto.

Infine, se nessuna delle due condizioni precedenti è verificata, significa che il cluster è composto sia da molecole d'acqua che di soluto. In questo caso si controlla il numero di molecole, se questo è maggiore di SOGLIA, viene incrementato l'ultimo elemento dell'array `Ws`, altrimenti viene incrementato l'elemento dell'array `Ws` il cui indice corrisponde al numero di molecole.
Il contenuto degli array viene poi scritto nel file delle frequenze.

2.3.5 Analisi del decadimento dei legami in funzione del tempo

Nella seconda parte del progetto si è studiato il decadimento nel tempo dei legami a idrogeno in una soluzione data.
Innanzi tutto viene creato un array di puntatori a grafo composto da n elementi. Poi viene creato un grafo vuoto per ciascuno degli n puntatori. Utilizzando le strutture dati e gli algoritmi descritti nelle sezioni precedenti, nel grafo i-esimo sono caricati i dati dell'i-esimo file PDB e sono poi individuati i relativi HBs.
Viene poi creato un array `DecadimentoLegami` di interi il cui elemento i-esimo memorizzerà il numero dei legami del grafo i-esimo che erano presenti anche nel grafo iniziale. Ovviamente, nel primo elemento dell'array viene inserito il numero complessivo dei legami del primo grafo. In seguito viene effettuata una scansione della lista delle molecole del primo grafo, per ogni molecola viene scandita la lista dei legami e per ogni legame si controlla se esso si "conserva" nei grafi successivi. Per verificare che un legame è presente nel grafo i-esimo, si controlla che in esso esista un legame avente accettore, donore e idrogeno con gli stessi indici del legame di cui si sta verificando la conservazione. In questo caso si incrementano gli elementi dell'array corrispondenti ai grafi in cui esiste il legame corrente. In conclusione nel file di output viene stampato il contenuto dell'array `DecadimentoLegami`.
Data la necessità di lavorare su più file di input, è stato deciso di imporre una struttura fissa (composta da tre parti separate da punti) al nome che i file PDB devono avere. Un esempio del nome di un file è il seguente:

```
urea.1.pdb
```

La prima parte del nome (urea) è comune a tutti i file di input, la seconda parte (1) è un numero progressivo che distingue univocamente i file in ordine di tempo e la terza è l'estensione che identifica il formato. Quando si chiama l'eseguibile vengono specificate come parametri la prima parte del nome dei file di input e il numero n dei file da caricare.
In seguito si è ristretta l'analisi alla sola prima shell. Fanno parte della prima shell tutte le molecole d'acqua che distano non più di 3.5 Å dalla molecola di soluto.
Per individuare queste molecole è stata utilizzata una funzione che verifica se l'ossigeno della molecola d'acqua dista al massimo 3.5 Å da un qualsiasi atomo della molecola di soluto. In questo

caso la molecola si trova in prima shell e vengono utilizzati gli algoritmi descritti in precedenza per vedere se i legami che essa forma si conservano negli snapshot successivi.

2.3.6 Analisi della distribuzione delle molecole d'acqua in funzione del numero degli HBs

Infine, è stata analizzata la distribuzione delle molecole di acqua in funzione del numero dei legami a idrogeno. In particolare è stato aggiunto, alla struttura dati della molecola, un contatore del numero dei legami a idrogeno che interessano la molecola. Utilizzando questo contatore si è visto, per ogni snapshot, quante molecole d'acqua erano isolate e quante formavano uno, due, tre, quattro o cinque legami.

Per caricare i dati e per stabilire gli HBs sono state utilizzate le stesse strutture dati e gli stessi algoritmi del paragrafo precedente.

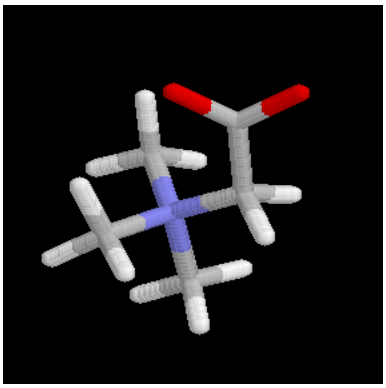
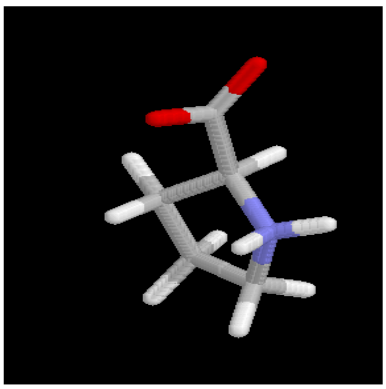
In seguito anche per questa analisi, si è voluto vedere il comportamento riguardante la prima shell.

3. Risultati ottenuti

In questo capitolo vengono descritte le molecole analizzate e sono mostrati e commentati i risultati che sono stati ottenuti eseguendo gli algoritmi citati nel capitolo precedente.

3.1 Descrizione molecole

Le molecole che sono state analizzate sono tutte osmoliti tranne l'urea. Gli osmoliti sono sostanze che influenzano l'osmosi. Il termine osmosi indica, in chimica e in fisica, il fenomeno consistente nel movimento di diffusione di due liquidi miscibili di diversa concentrazione, attraverso un setto poroso o una membrana, semipermeabile o permeabile ai due mezzi. Queste sostanze, quindi, proteggono molecole e cellule contro la presenza di alta concentrazione di sali nell'ambiente circostante.

 **Figura 3.1 – Molecola di GB**	La GB (Glycine betaine) è uno degli osmoprotettori più studiati ed efficienti. È presente non solo nei batteri, ma anche nelle cellule eucariotiche [7].
 **Figura 3.2 – Molecola di prolina**	La prolina è un amminoacido apolare. La sua molecola è chirale, cioè non sovrapponibile alla sua immagine speculare. Si presenta come un solido cristallino bianco [8].

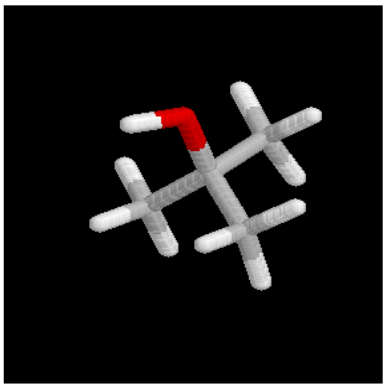


Figura 3.3 – Molecola di TBA

La molecola di alcool tert-butilico (TBA) è un denaturante delle proteine particolarmente efficace, caratterizzato dalla più larga parte idrofobica tra gli alcool idrosolubili [9].

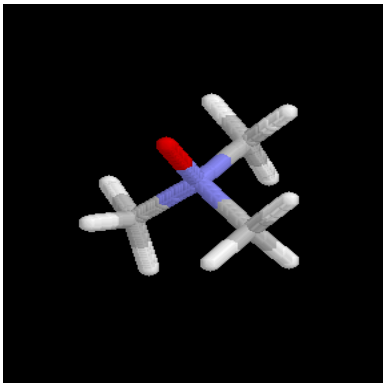


Figura 3.4 – Molecola di TMAO

Il TMAO (trimetilamina N-ossido) è una molecola che agisce come regolatore della pressione osmotica nei fluidi intracellulari [10].

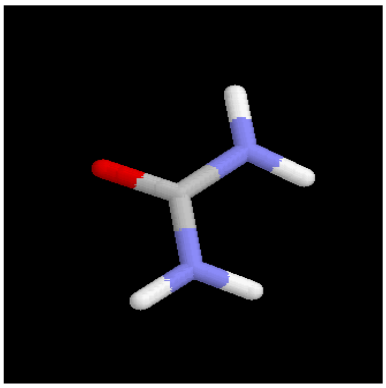


Figura 3.5 – Molecola di urea

L'urea è il prodotto finale della degradazione metabolica delle proteine. Si presenta come un composto cristallino incolore. Ad alta concentrazione è un denaturante di proteine [5].

Legenda:

	Ossigeno
	Idrogeno
	Azoto
	Carbonio

3.2 Statistiche sui cluster

I valori riportati nei grafici (Figure 3.6) sono valori normalizzati e rappresentano, in percentuale, il numero di molecole che compongono i cluster. L' istogramma "TOTALE" rappresenta tutti i cluster della soluzione, "SOLO ACQUA" rappresenta i cluster composti da sole molecole d'acqua, "SIA ACQUA CHE SOLUTO" rappresenta i cluster composti sia da molecole d'acqua sia da molecole di soluto, e infine nella tabella "SOLO SOLUTO" sono rappresentati i cluster composti da sole molecole di soluto.

3.2.1 Comportamento di ogni singolo soluto

Per ogni soluto vengono di seguito rappresentati i grafici "TOTALE", "SOLO ACQUA", "SIA ACQUA CHE SOLUTO", ad eccezione del TBA, per il quale è riportato anche il grafico "SOLO SOLUTO", in quanto, a differenza delle altre sostanze, si comporta in modo anomalo. Si può notare, infatti, che in una singola snapshot la molecola di TBA viene isolata dal resto della soluzione formando un "cluster" a sé.

In ascissa viene indicato il numero di molecole di cui sono composti i cluster, mentre nelle ordinate è riportata la frequenza (espressa tra 0 e 1) di cluster aventi un dato numero di molecole.

- **UREA**

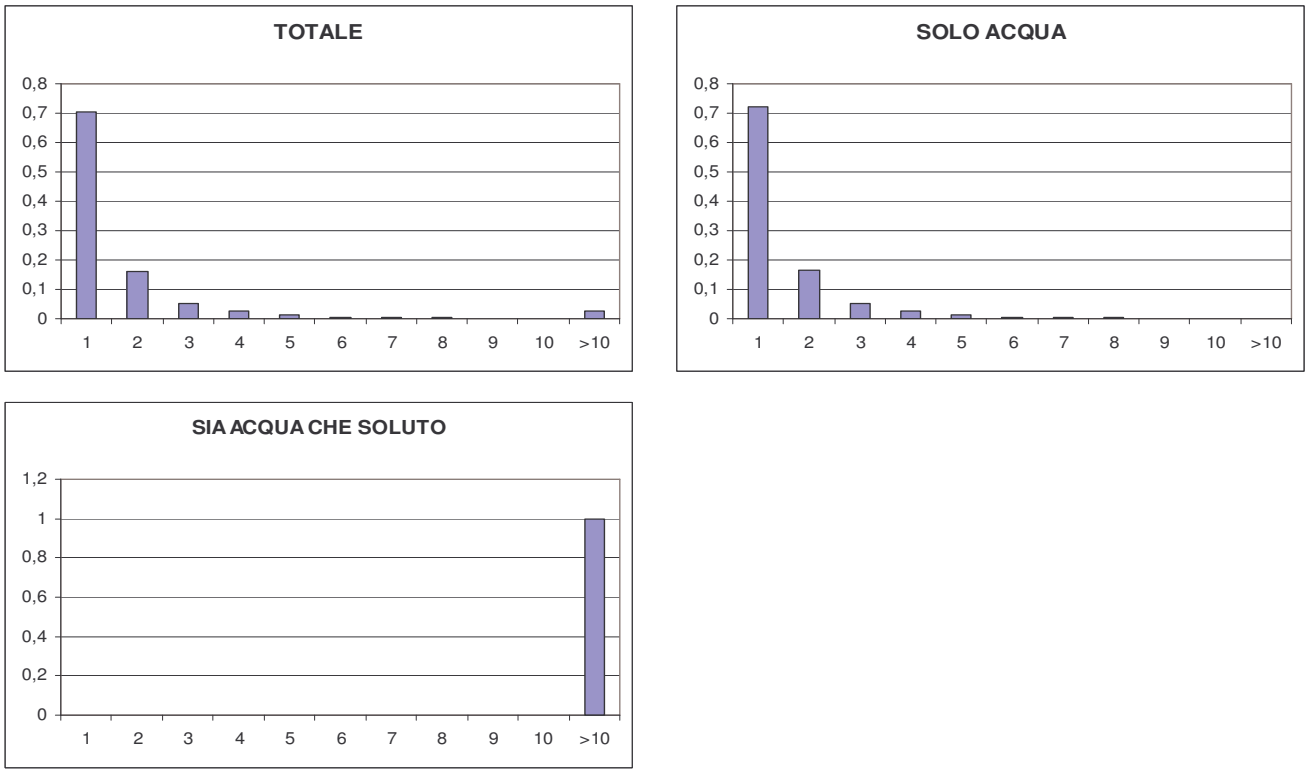


Figura 3.6 a – Distribuzione percentuale dei cluster in funzione del numero di molecole di cui sono composti, in soluzione acquosa con una molecola di urea.

- **PROLINA**

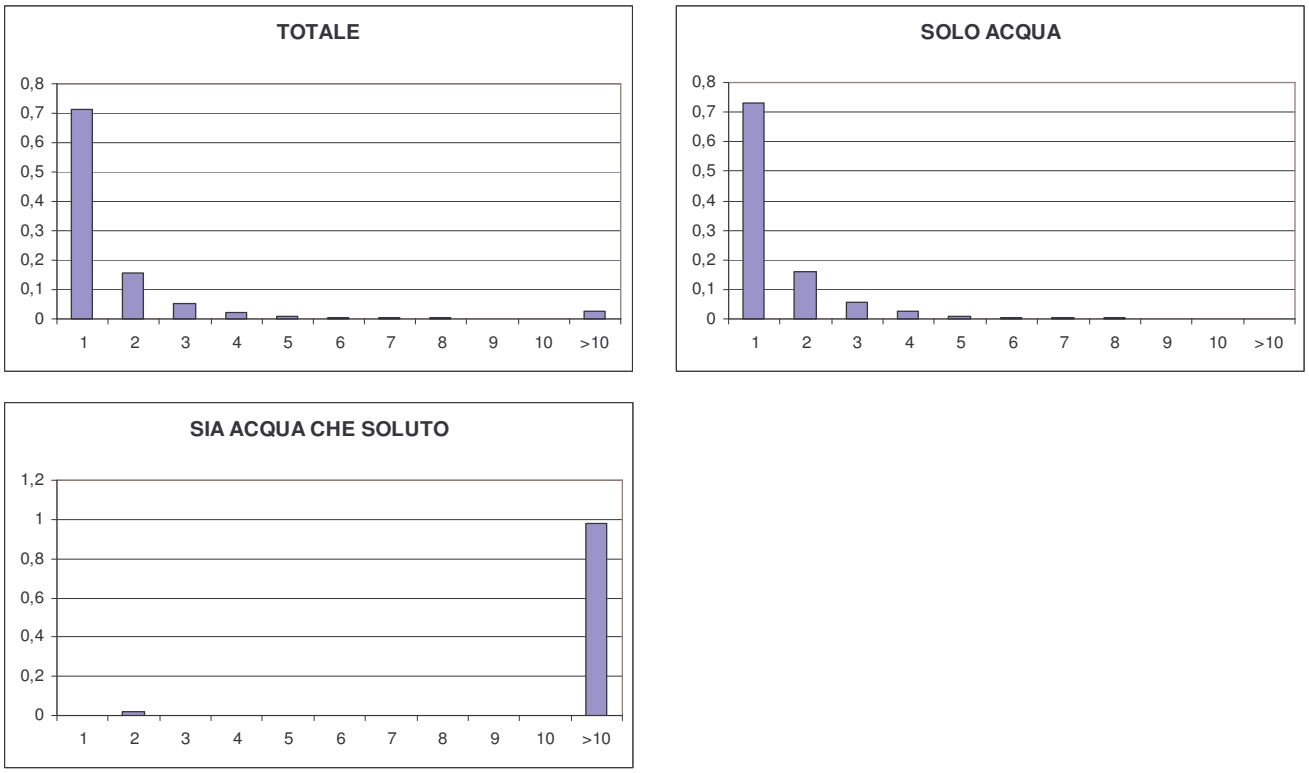


Figura 3.6 b - Distribuzione percentuale dei cluster in funzione del numero di molecole di cui sono composti, in soluzione acquosa con una molecola di prolina

- **TBA**

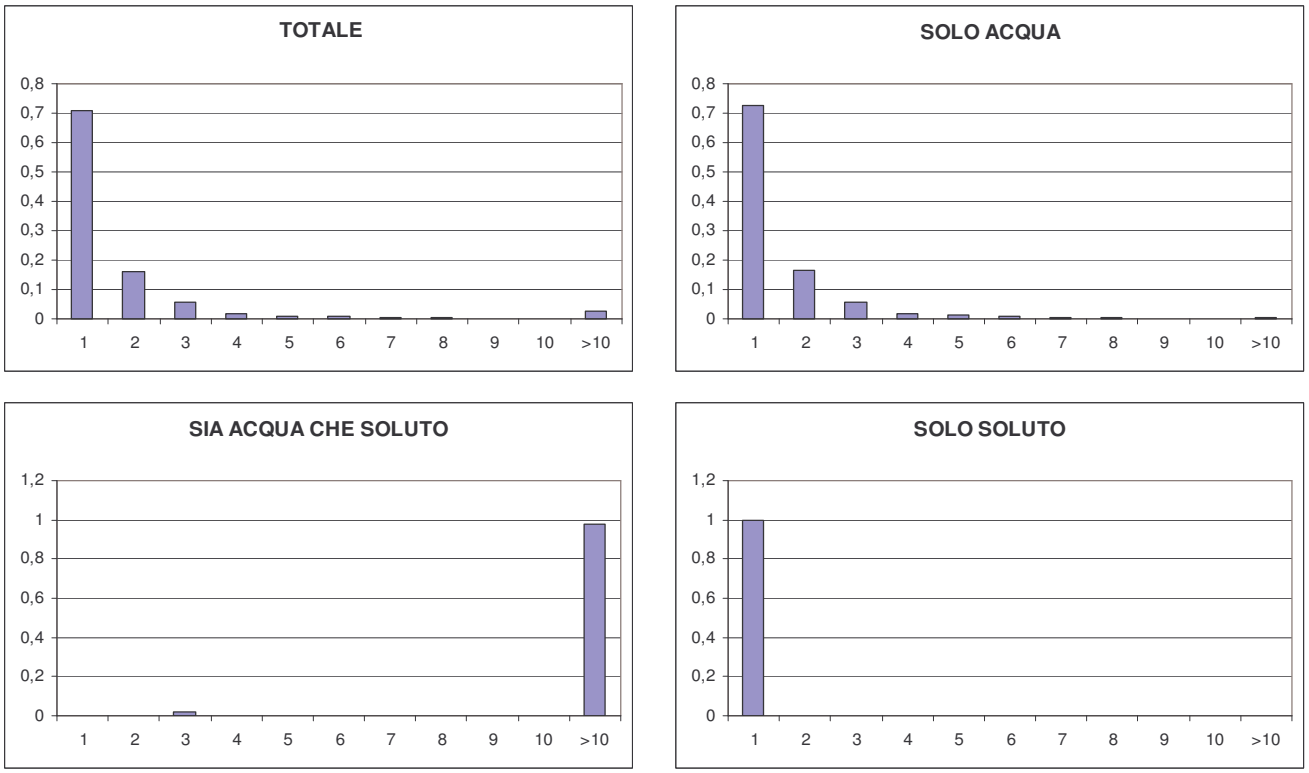


Figura 3.6 c - Distribuzione percentuale dei cluster in funzione del numero di molecole di cui sono composti, in soluzione acquosa con una molecola di TBA

- **GB**

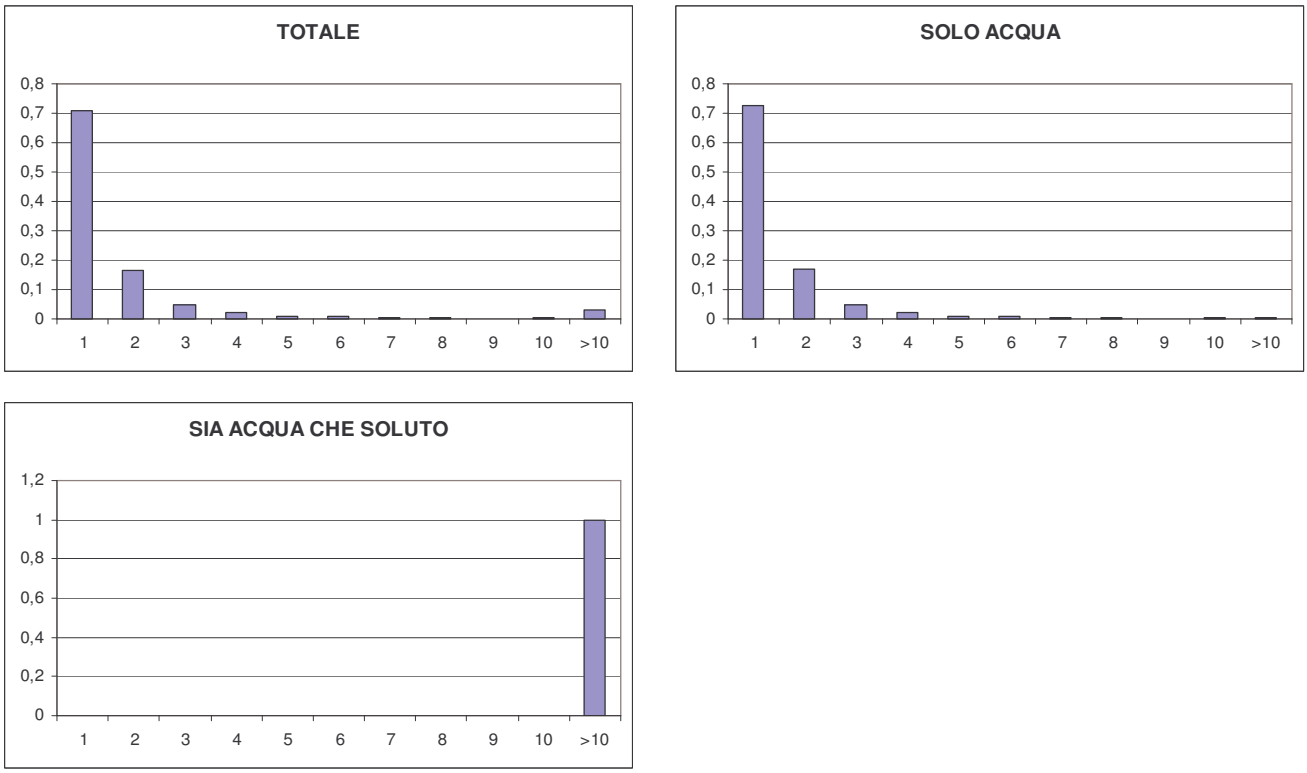


Figura 3.6 d - Distribuzione percentuale dei cluster in funzione del numero di molecole di cui sono composti, in soluzione acquosa con una molecola di GB

- **TMAO**

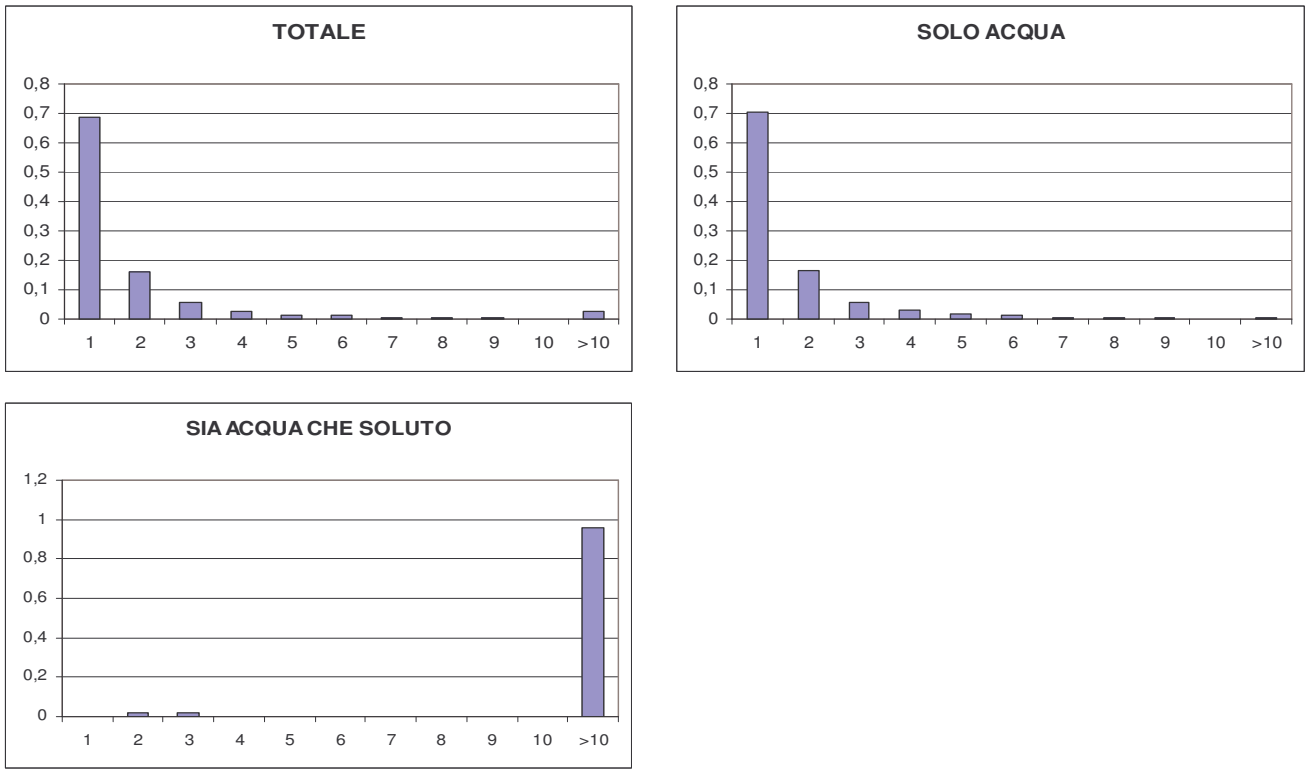


Figura 3.6 e - Distribuzione percentuale dei cluster in funzione del numero di molecole di cui sono composti, in soluzione acquosa con una molecola di TMAO

3.2.2 Confronto tra soluti

Dai grafici in Figura 3.7, che mostrano il confronto sulla distribuzione dei cluster in base al numero di molecole, si può notare che le sostanze si comportano in modo abbastanza simile. Si vede che la maggior parte dei cluster sono costituiti da una sola molecola d'acqua isolata. In tutti e cinque i casi si forma un cluster di grandi dimensioni, contenente la molecola di soluto. È possibile, inoltre, notare il comportamento particolare della TBA dal grafico che mostra i cluster composti da sole molecole di soluto. In una snapshot, si verifica che un cluster contiene solo la molecola di TBA; questa situazione potrebbe verificarsi a causa della parte fobica di questo soluto. Infatti gli oggetti idrofobici favoriscono la strutturazione dell'acqua, e in questo caso la parte fobica del TBA aiuta le molecole d'acqua a formare legami tra di esse tendendo ad isolare la molecola di TBA.

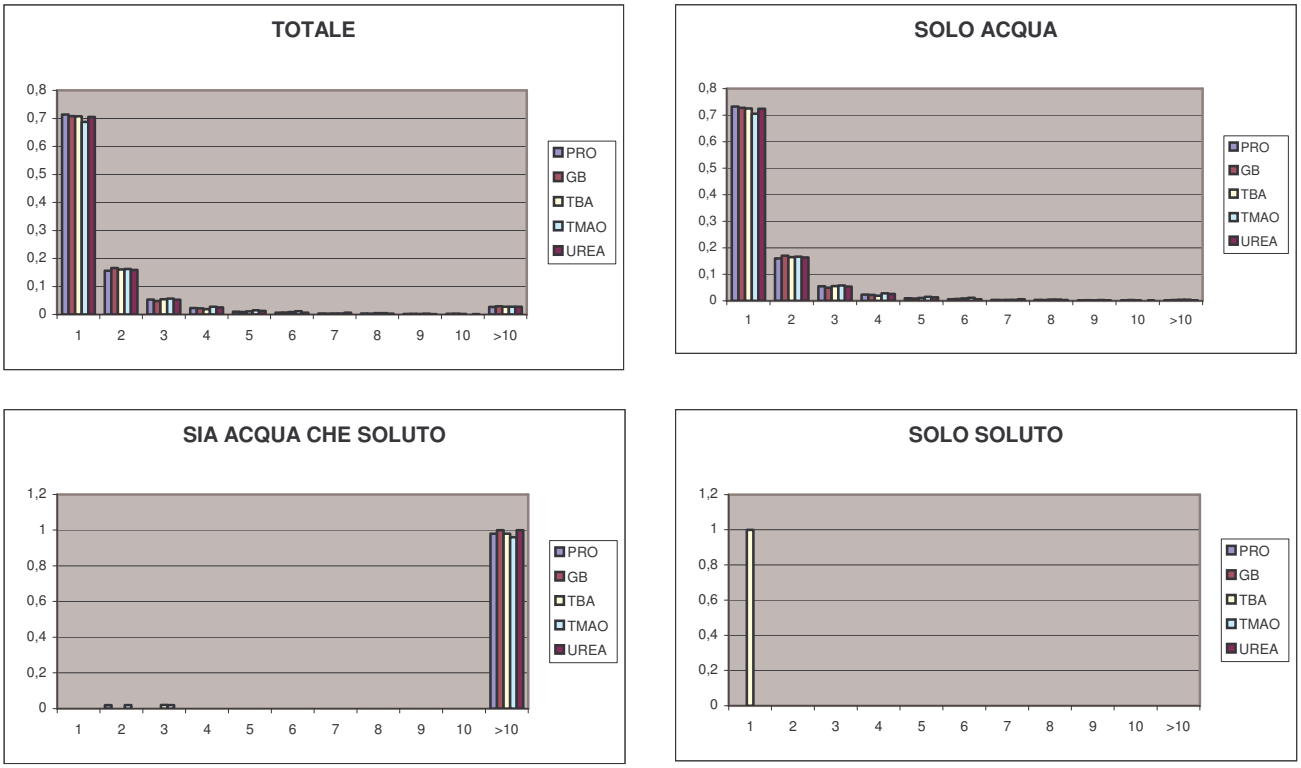


Figura 3.7 – Confronto della distribuzione percentuale dei cluster in funzione del numero di molecole di cui sono composti.

Di seguito sono riportate le tabelle con i valori da cui sono stati ricavati i grafici in figura 3.7

Tabella 3.1 – Valori normalizzati che rappresentano la distribuzione percentuale dei cluster in funzione del numero di molecole di cui sono composti.

TOTALE

	1	2	3	4	5	6	7	8	9	10	>10
PRO	0,714214	0,15625	0,053427	0,023185	0,0095766	0,006048	0,003528	0,003528	0,001512	0,002016	0,026714
GB	0,708312	0,165708	0,048092	0,021955	0,0083638	0,006796	0,003136	0,003136	0,002091	0,003136	0,029273
TBA	0,707821	0,160846	0,054599	0,018692	0,0108214	0,008854	0,003443	0,004427	0,001476	0,001476	0,027545
TMAO	0,687624	0,162871	0,056931	0,027723	0,0148515	0,011386	0,003465	0,004455	0,00297	0	0,027723
UREA	0,705252	0,159102	0,052524	0,025497	0,0127486	0,006119	0,006119	0,00306	0,00102	0,00102	0,027537

SOLO ACQUA

	1	2	3	4	5	6	7	8	9	10	>10
PRO	0,732678	0,159772	0,054809	0,023785	0,0098242	0,006205	0,003619	0,003619	0,001551	0,002068	0,002068
GB	0,727322	0,170156	0,049383	0,022544	0,0085883	0,006978	0,003221	0,003221	0,002147	0,003221	0,003221
TBA	0,725164	0,164902	0,055472	0,019163	0,0110943	0,009077	0,00353	0,004539	0,001513	0,001513	0,004034
TMAO	0,705076	0,166497	0,057868	0,028426	0,0152284	0,011675	0,003553	0,004569	0,003046	0	0,004061
UREA	0,723705	0,163265	0,053898	0,026164	0,0130822	0,006279	0,006279	0,00314	0,001047	0,001047	0,002093

SIA ACQUA CHE SOLUTO

	1	2	3	4	5	6	7	8	9	10	>10
PRO	0	0,02	0	0	0	0	0	0	0	0	0,98
GB	0	0	0	0	0	0	0	0	0	0	1
TBA	0	0	0,020408	0	0	0	0	0	0	0	0,979592
TMAO	0	0,02	0,02	0	0	0	0	0	0	0	0,96
UREA	0	0	0	0	0	0	0	0	0	0	1

SOLO SOLUTO

	1	2	3	4	5	6	7	8	9	10	>10
PRO	0	0	0	0	0	0	0	0	0	0	0
GB	0	0	0	0	0	0	0	0	0	0	0
TBA	1	0	0	0	0	0	0	0	0	0	0
TMAO	0	0	0	0	0	0	0	0	0	0	0
UREA	0	0	0	0	0	0	0	0	0	0	0

3.3 Decadimento dei legami

Nelle figure 3.8a e 3.8b sono riportati i grafici inerenti al decadimento dei legami a idrogeno che si conservano nel tempo. Sono stati analizzati i legami che fanno parte di tutta la soluzione e quelli in prima shell.

I grafici sono in scala logaritmica. In ascissa è riportato il tempo in picosecondi, in ordinata il logaritmo, rispetto ai valori normalizzati.

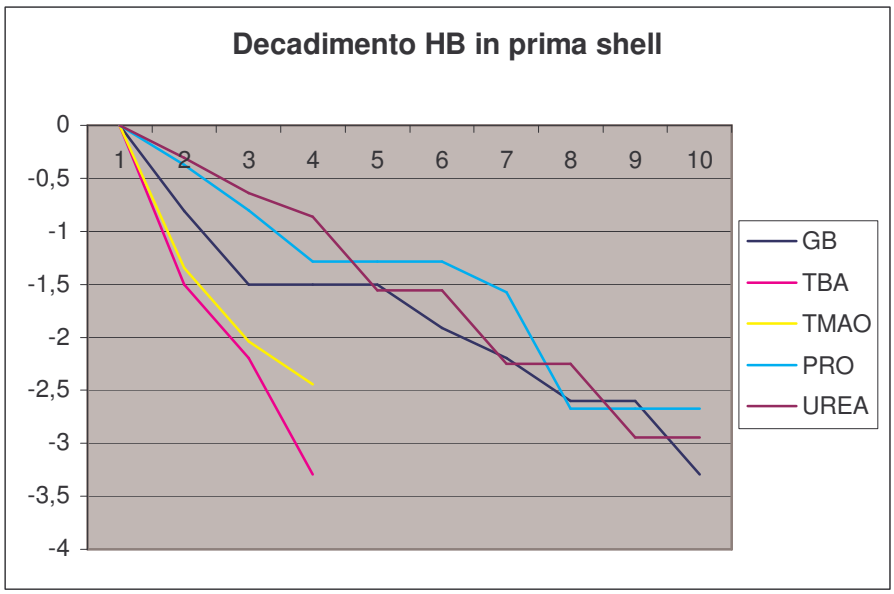


Figura 3.8 a - Decadimento legami a idrogeno in tutta la soluzione

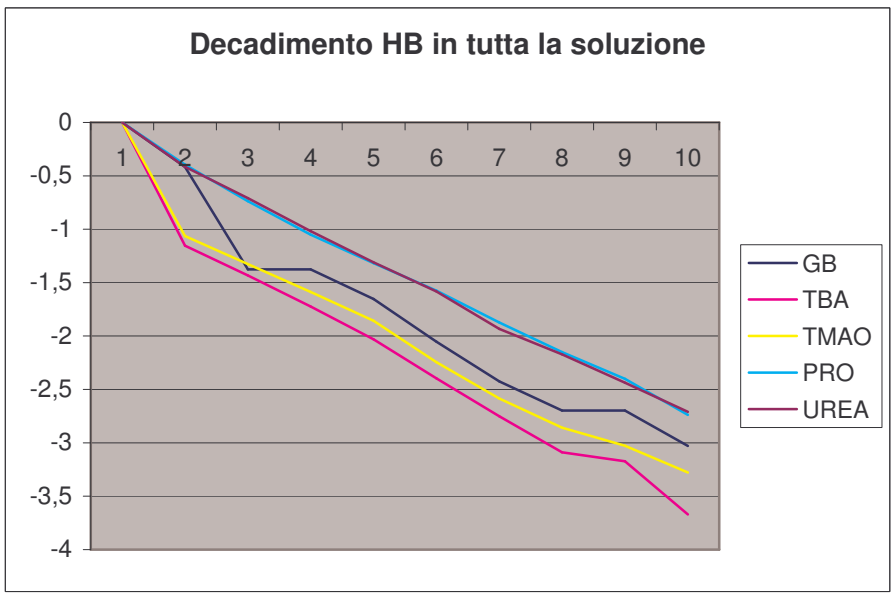


Figura 3.8 b - Decadimento legami a idrogeno in prima shell

Dal grafico in Figura 3.8a, che rappresenta il decadimento degli HBs su tutta la soluzione, si può notare che prolina e urea hanno un andamento molto simile. Probabilmente questo comportamento è dovuto al fatto che entrambe le molecole hanno gruppi NH, con cui possono formare HBs come donori e quindi decadono più lentamente.

Al contrario, TMAO e GB non hanno modo di formare HBs donori, quindi decadono più velocemente. I legami a idrogeno nella soluzione contenente TBA decadono più velocemente seguendo un comportamento simile a quello del TMAO.

Dal grafico, in Figura 3.8 b, che rappresenta il decadimento degli HBs in prima shell, si può ancora notare che prolina e urea hanno un comportamento simile, le linee che rappresentano l'andamento del decadimento tendono ad incrociarsi. TBA e TMAO tendono a decadere molto velocemente, mentre i legami intorno alla GB inizialmente decadono velocemente, in seguito si ha un rallentamento del decadimento, infatti la linea che la rappresenta va ad incrociarsi con quelle inerenti a prolina e urea.

Nelle due tabelle sottostanti (tabelle 3.2) sono riportati i valori normalizzati tra 0 e 1 dei legami che si conservano per ciascuna sostanza presa in considerazione.

Tabella 3.2 a - Valori normalizzati del numero dei legami che si conservano in tutta la soluzione

Tutte le molecole				
GB	TBA	TMAO	PRO	UREA
1	1	1	1	1
0,658947	0,315118	0,344704	0,672828	0,659696
0,252632	0,238616	0,265709	0,478743	0,492395
0,252632	0,178506	0,204668	0,349353	0,361217
0,191579	0,131148	0,156194	0,268022	0,269962
0,128421	0,091075	0,105925	0,207024	0,205323
0,088421	0,063752	0,075404	0,15342	0,144487
0,067368	0,045537	0,057451	0,116451	0,114068
0,067368	0,041894	0,048474	0,090573	0,087452
0,048421	0,025501	0,037702	0,064695	0,06654

Tabella 3.2 b - Valori normalizzati del numero dei legami che si conservano in prima shell

Prima Shell				
GB	TBA	TMAO	PRO	UREA
1	1	1	1	1
0,444444	0,222222	0,26087	0,689655	0,736842
0,222222	0,111111	0,130435	0,448276	0,526316
0,222222	0,037037	0,086957	0,275862	0,421053
0,222222	0	0	0,275862	0,210526
0,148148	0	0	0,275862	0,210526
0,111111	0	0	0,206897	0,105263
0,074074	0	0	0,068966	0,105263
0,074074	0	0	0,068966	0,052632
0,037037	0	0	0,068966	0,052632

Infine, nelle due tabelle successive (tabelle 3.3) sono riportati i logaritmi dei valori normalizzati presenti nelle tabelle precedenti.

Tabella 3.3 a - Logaritmi dei valori normalizzati degli HBs che si conservano in tutta la soluzione

Tutte le molecole				
GB	TBA	TMAO	PRO	UREA
0	0	0	0	0
-0,41711	-1,15481	-1,06507	-0,39627	-0,41598
-1,37582	-1,4329	-1,32535	-0,73659	-0,70847
-1,37582	-1,72313	-1,58637	-1,05167	-1,01828
-1,65246	-2,03143	-1,85666	-1,31669	-1,30947
-2,05244	-2,39608	-2,24503	-1,57492	-1,58317
-2,42565	-2,75275	-2,5849	-1,87458	-1,93457
-2,69758	-3,08922	-2,85683	-2,15028	-2,17096
-2,69758	-3,1726	-3,02673	-2,4016	-2,43666
-3,02782	-3,66904	-3,27804	-2,73807	-2,70995

Tabella 3.3 b - Logaritmi dei valori normalizzati degli HBs che si conservano in prima shell

Prima Shell				
GB	TBA	TMAO	PRO	UREA
0	0	0	0	0
-0,81093	-1,50408	-1,34373	-0,37156	-0,30538
-1,50408	-2,19722	-2,03688	-0,80235	-0,64185
-1,50408	-3,29584	-2,44235	-1,28785	-0,865
-1,50408			-1,28785	-1,55814
-1,90954			-1,28785	-1,55814
-2,19722			-1,57554	-2,25129
-2,60269			-2,67415	-2,25129
-2,60269			-2,67415	-2,94444
-3,29584			-2,67415	-2,94444

3.4 Tempo di decadimento (τ)

L'andamento del decadimento nel tempo può essere approssimato con la funzione $y = Ae^{-\frac{t}{\tau}}$. Il valore τ descrive il tempo di decadimento.

Dato che nelle figure 3.8 il decadimento, in scala logaritmica, ha un andamento piuttosto lineare, è stato utilizzato il Best Fit Lineare per trovare la pendenza di ciascuna funzione che rappresenta il decadimento dei legami. I valori di τ, in tabella 3.4, sono dati dall'inverso della pendenza.

Tabella 3.4 – tempo di decadimento (τ) in prima shell e nell'intera soluzione

	Prima shell	Intera soluzione
GB	-3,33	-3,13
TBA	-0,94	-2,78
TMAO	-1,25	-3,03
PRO	-3,33	-3,45
UREA	-2,78	-3,33

Osservando le tabelle di ciascuna sostanza, si può notare che, Prolina e GB hanno lo stesso τ, e simile è anche quello dell'urea. Queste tre molecole si comportano in modo affine e i loro legami decadono in circa 3 picosecondi. TBA e TMAO, invece, hanno un τ che vale circa un picosecondo.

3.5 Distribuzione delle molecole d'acqua in funzione del numero di HBs

Le tabelle seguenti rappresentano la distribuzione delle molecole d'acqua in funzione dei legami a idrogeno che esse formano. I valori sono calcolati come media su tutti i 50 snapshot che sono stati analizzati, e sono riportati come frequenza espressa tra 0 e 1. La prima tabella di ogni soluto mostra i valori degli HBs in tutta la soluzione, la seconda tabella invece mostra gli HBs in prima shell.

3.5.1 Comportamento di ogni singolo soluto

- **UREA**

Tutte le molecole

Tabella 3.5 a - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di urea.

	0	1	2	3	4	5
Tot. Norm.	0,055102	0,221682	0,344595	0,268696	0,102514	0,007411

Molecole in prima shell

Tabella 3.5 b - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di urea.

	0	1	2	3	4	5
Tot. Norm.	0,012195	0,121951	0,271341	0,38872	0,192073	0,01372

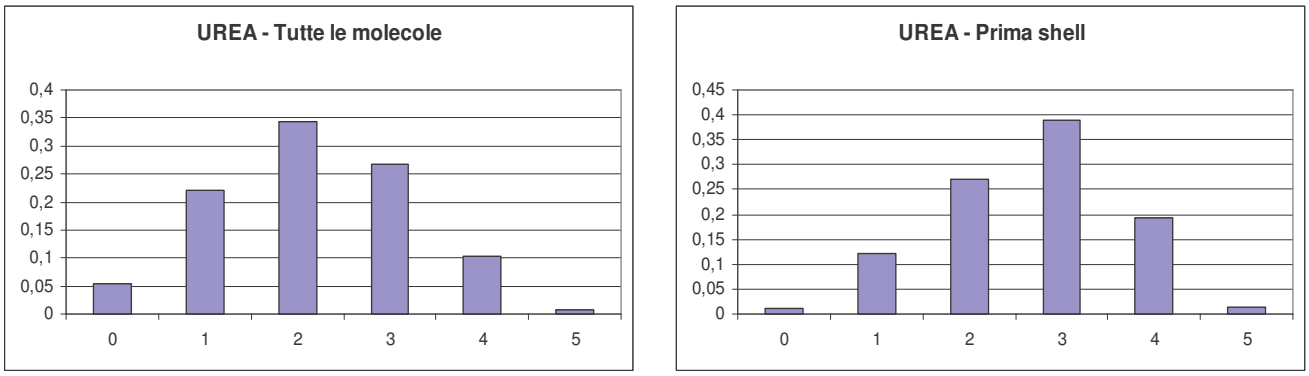


Figura 3.9 a - Distribuzione delle molecole d'acqua in funzione del numero di legami a idrogeno che formano, in soluzione con una molecola di urea

- **PROLINA**

Tutte le molecole

Tabella 3.6 a - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di prolina.

	0	1	2	3	4	5
Tot. Norm.	0,056353	0,218055	0,343448	0,27413	0,100378	0,007636

Molecole in prima shell

Tabella 3.5 b - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di prolina.

	0	1	2	3	4	5
Tot. Norm.	0,025641	0,117521	0,360043	0,344017	0,141026	0,011752

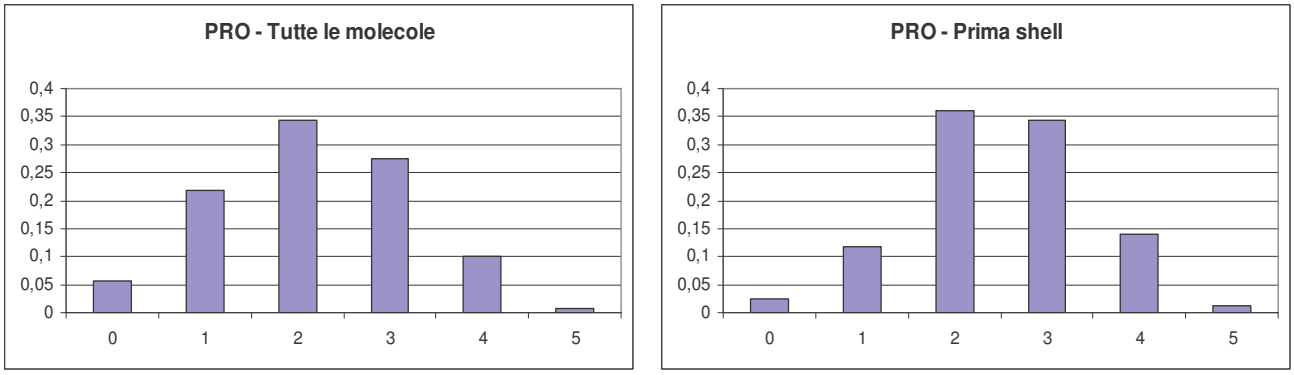


Figura 3.9 b - Distribuzione delle molecole d'acqua in funzione del numero di legami a idrogeno che formano, in soluzione con una molecola di prolina.

- **TBA**

Tutte le molecole

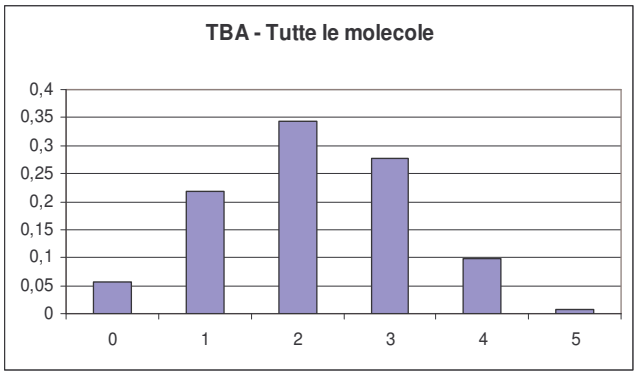
Tabella 3.7 a - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di TBA.

	0	1	2	3	4	5
Tot. Norm.	0,057068	0,218708	0,342607	0,27657	0,098024	0,007024

Molecole in prima shell

Tabella 3.7 b - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di TBA.

	0	1	2	3	4	5
Tot. Norm.	0,014667	0,129333	0,317333	0,372	0,16	0,006667



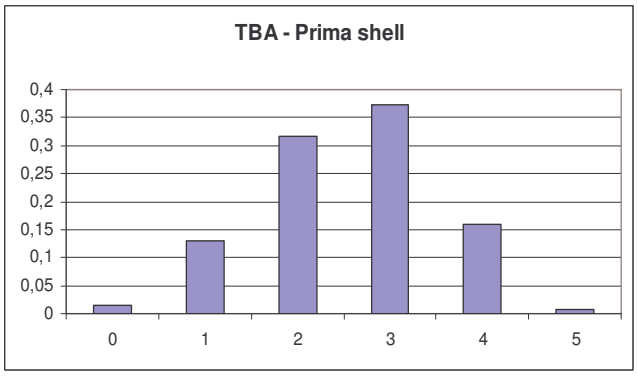


Figura 3.9 c - Distribuzione delle molecole d'acqua in funzione del numero di legami a idrogeno che formano, in soluzione con una molecola di TBA

- **GB**

Tutte le molecole

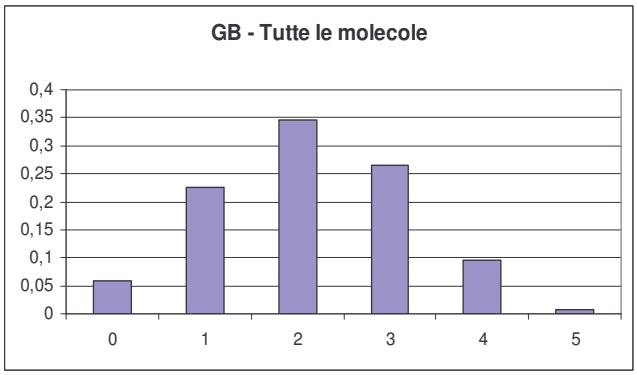
Tabella 3.8 a - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formno, in una soluzione con una molecola di GB.

	0	1	2	3	4	5
Tot. Norm.	0,059823	0,225298	0,345254	0,265607	0,095894	0,008124

Molecole in prima shell

Tabella 3.8 b - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di GB.

	0	1	2	3	4	5
Tot. Norm.	0,015595	0,104288	0,332359	0,380117	0,15692	0,010721



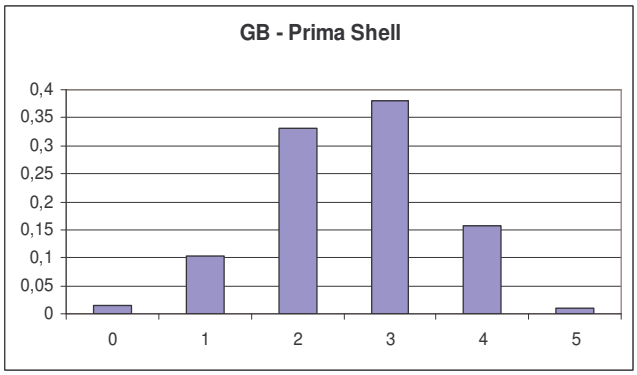


Figura 3.9 d - Distribuzione delle molecole d'acqua in funzione del numero di legami a idrogeno che formano, in soluzione con una molecola di GB

- **TMAO**

Tutte le molecole

Tabella 3.9 a - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di TMAO.

	0	1	2	3	4	5
Tot. Norm.	0,055123	0,229939	0,337527	0,266767	0,103897	0,006747

Molecole in prima shell

Tabella 3.9 b - Valori normalizzati della distribuzione delle molecole d'acqua in tutta la soluzione, in funzione del numero di HBs che formano, in una soluzione con una molecola di TMAO.

	0	1	2	3	4	5
Tot. Norm.	0,020507	0,124246	0,296743	0,37877	0,168878	0,010856

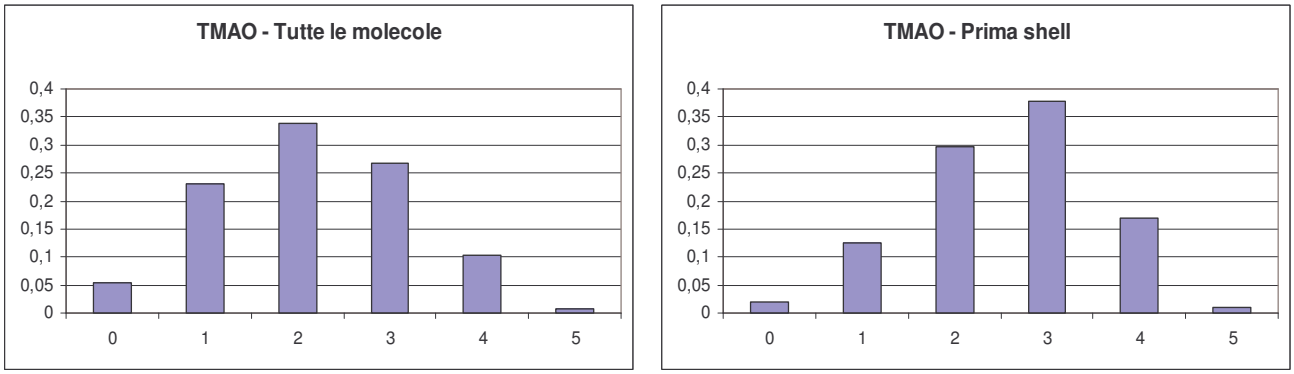


Figura 3.9 e - Distribuzione delle molecole d'acqua in funzione del numero di legami a idrogeno che formano, in soluzione con una molecola di TMAO

Dai grafici (Figure 3.9) si può notare che, considerando tutta la soluzione, le molecole d'acqua tendono a formare principalmente 2 HBs, e una buona parte forma 1 e 3 HBs. Invece, se si considera solo la prima shell, le molecole d'acqua tendono a formare principalmente 3 legami a idrogeno, diminuisce il numero di molecole che formano solo 2 legami, mentre aumenta il numero di molecole d'acqua che formano 4 legami.

3.5.2 Confronto tra soluti

Tutte le molecole

Tabella 3.10 a - Confronto tra i valori normalizzati che rappresentano la distribuzione delle molecole d'acqua in tutta la soluzione in funzione del numero di HBs che formano

	0	1	2	3	4	5
PRO	0,056353	0,218055	0,343448	0,27413	0,100378	0,007636
GB	0,059823	0,225298	0,345254	0,265607	0,095894	0,008124
TBA	0,057068	0,218708	0,342607	0,27657	0,098024	0,007024
TMAO	0,055123	0,229939	0,337527	0,266767	0,103897	0,006747
UREA	0,055102	0,221682	0,344595	0,268696	0,102514	0,007411

Molecole in prima shell

Tabella 3.10 a - Confronto tra i valori normalizzati che rappresentano la distribuzione delle molecole d'acqua in prima shell in funzione del numero di HBs che formano

	0	1	2	3	4	5
PRO	0,025641	0,117521	0,360043	0,344017	0,141026	0,011752
GB	0,015595	0,104288	0,332359	0,380117	0,15692	0,010721
TBA	0,014667	0,129333	0,317333	0,372	0,16	0,006667
TMAO	0,020507	0,124246	0,296743	0,37877	0,168878	0,010856
UREA	0,012195	0,121951	0,271341	0,38872	0,192073	0,01372

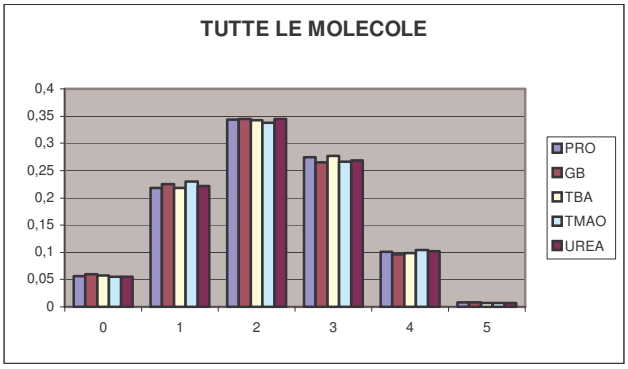


Figura 3.10 a – Confronto della distribuzione delle molecole d'acqua in funzione dei legami a idrogeno in tutta la soluzione

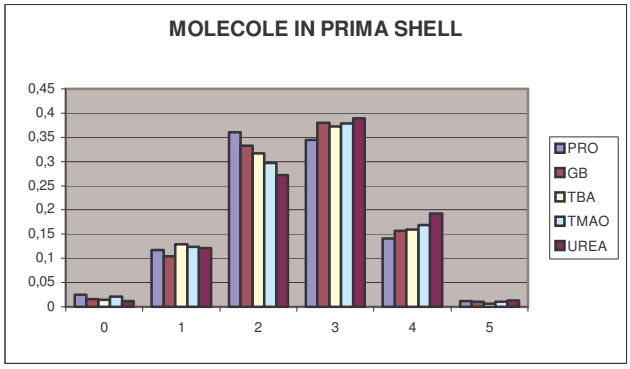


Figura 3.10 b – Confronto della distribuzione delle molecole d'acqua in funzione dei legami a idrogeno in prima shell

Dal confronto (Figure 3.10) si vede ancora meglio che le molecole d'acqua, considerando solo la prima shell piuttosto che l'intera soluzione, tendano maggiormente a formare tre legami a idrogeno. Il numero di molecole che formano quattro legami aumenta soprattutto nella soluzione contenente urea, dove quasi raddoppia. Dal confronto è strano notare che il TBA e il TMAO abbiano un andamento simile, tenendo conto della caratteristica del TBA, che ad alta concentrazione tende a denaturare le proteine.

Appendice – Codice sorgente

Di seguito sono riportate le parti significative del codice sorgente, in linguaggio C, utilizzato per implementare gli algoritmi descritti nel capitolo 2, impiegati per effettuare le analisi.

Strutture dati

```
struct _Grafo {
  int   NumMolecole;
  int   NumLegami;

  ListaMolecole * LM;
 };

struct _Molecola {
    int Indice;
    char Nome[3];
    ListaLegami *Legami;  //Lista dei legami H che interessano la molecola
    ListaAtomi * AccDon; //Lista degli atomi donori/accettori che compongono la
                         // molecola
    ListaAtomi * Idrogeni;
    ListaAtomi * Altri; // Lista degli altri atomi che compongono la molecola
    Molecola *Succ, *Prec;
    boolean Visitato;
     int NumHB;    //Numero dei legami ad idrogeno che interessano la molecola
};

struct _Legame {
    int Indice;
    float Distanza;
    float Angolo;
    Atomo *Acc, *Don, *H;
    Legame *Prec, *Succ, *Opposto;
    Molecola *M1, *M2;
};

struct _Atomo  {
 //elemento della lista degli atomi che compongono una molecola
  int  Indice;
  float  x;
  float  y;
  float z;
  char Elemento[4];
  Molecola *M;
  Atomo *Succ, *Prec;
};

struct _pMolecola {
    Molecola *Mol;
    pMolecola *Succ, *Prec;
};
```

Statistiche sui cluster

Funzione Main

```
int main(int argc, char *argv[])
{
  char InputFile[LUNGHEZZA];  //Nome del file pdb di input
  char OutputFile[LUNGHEZZA]; //Nome del file pdb di output
  char FrequencyFile[LUNGHEZZA];    //Nome del file cui si appendono le
                                    //frequenze dei cluster
  char LogFile[LUNGHEZZA]; //Nome del file che conterrà il log (elenco delle
                           //molecole, legami, cluster...) del programma
  float ALFA, DELTA;
  int SOGLIA;
  Grafo *G;

  //Legge il file di configurazione
  LeggeConfigurazione(&ALFA, &DELTA, &SOGLIA);
  // Interpretazione della linea di comando
  LeggeInput(argc, argv,InputFile, OutputFile, FrequencyFile);
  //Completa la scrittura del nome del file di log nel formato
  //nomefileinput.log.txt
  strcpy(LogFile, InputFile);
  strcat(LogFile, ".log.txt");
  // Crea un grafo vuoto
  G = CreaGrafo();
  // Carica i dati di ogni atomo del file di input nel grafo G
  printf("Attendi, sto creando il grafo\n");
  CaricaAtomi(InputFile,G);
  // Stabilisce i legami del grafo
  printf("Attendi, sto creando i legami\n");
  StabilisciLegamiGrafo(G,ALFA,DELTA, InputFile);
  StabilisciCluster(G);
  FrequenzaCluster(G,SOGLIA,FrequencyFile);
  //stampa il file pdb nel file di output specificato
  ScriviPdb(G, OutputFile);
  //Scrive sul file specificato diverse informazioni per il debugging
  StampaGrafo(G, LogFile);
  //Distrugge il grafo e le strutture che lo compongono
  DistruggiGrafo(G);
  // system("PAUSE");
  return 0;
}

Grafo *CreaGrafo ()
{
  Grafo *G;

  G = (Grafo *) malloc(sizeof(Grafo));
  if (G == NULL)   {
    printf("Memoria insufficiente per allocare il grafo!\n");
    exit(EXIT_MEMORY);
  }

  G->NumMolecole = 0;
  G->NumLegami = 0;
  G->NumCluster = 0;
  G->LM = (ListaMolecole *) CreaListaMolecole();
  G->LC = (ListaCluster *) CreaListaCluster();
  return G;
}
```

Lettura del file di configurazione

```
void LeggeConfigurazione(float *ALFA, float *DELTA, int *SOGLIA)
{
    FILE *fConfigFile;
    char Parametro[100];
    float Valore;

    fConfigFile = fopen("configurazione.txt","r");
    if (fConfigFile == NULL)
    {
    printf("Il file configurazione.txt non esiste!\n");
    exit(EXIT_OPENFILE);
    }

    while(!feof(fConfigFile))
    {
        fscanf(fConfigFile,"%s %f\n", Parametro, &Valore);

        if(Parametro[0]=='#')
              continue;
        else
        {
              if (!strcmp(Parametro,"ALFA"))
                 *ALFA=Valore;
              if (!strcmp(Parametro,"DELTA"))
                 *DELTA=Valore;
              if (!strcmp(Parametro,"SOGLIA"))
                 *SOGLIA=(int)Valore;
        }
    }

    fclose(fConfigFile);
}
```

Interpretazione della linea di comando

```
//Legge i parametri specificati da linea di comando e restituisce
//il nome del file di input in InputFile
void LeggeInput(int argc, char *argv[], char *InputFile, char *OutputFile, char
*FrequencyFile)
{
  if (argc != 4)
  {
    printf("Formato della linea di comando errato!\n");
      printf("Usare: %s file_di_input.pdb file_di_output.pdb
            file_frequenze\n",argv[0]);
    exit(EXIT_WRONGCOMMANDLINE);
  }

  strcpy(InputFile,argv[1]);
  strcpy(OutputFile,argv[2]);
  strcpy(FrequencyFile,argv[3]);
}
```

Lettura del file di input

```
//Legge gli atomi dal file di input e crea le strutture delle molecole nel grafo
void CaricaAtomi(char *InputFile, Grafo *G)
{
  //Variabili in cui si memorizzano i dati letti per ogni atomo
  int id_atomo, id_molecola;
  char tipo_atomo[4];
  char tipo_molecola[3];
  float x, y, z;
  FILE *fInputFile;
  Molecola *pM;  //punta alla molecola corrente nel grafo
  Atomo *pA;     //punta all'atomo corrente
  pM=NULL;
  pA=NULL;


  fInputFile = fopen(InputFile,"r");
  if (fInputFile == NULL)
  {
    printf("Il file non esiste!\n",InputFile);
    exit(EXIT_OPENFILE);
  }

  while(!feof(fInputFile))
  {
    if (fscanf(fInputFile,"ATOM %d %s %s %d %f %f %f 0.00 0.00\n",&id_atomo,
                tipo_atomo, tipo_molecola, &id_molecola, &x, &y, &z) != 7)
    {
      printf("Errore nel formato del file di input!\n");
      exit(EXIT_WRONGINPUTFORMAT);
    }
    else
    {
      /*il codice sottostante presuppone gli atomi del file di input siano
      ordinati per id crescente delle molecole*/
      if ((pM == NULL) || (id_molecola != pM->Indice)) //Se la molecola
                                                       //dell'atomo non esiste
                                                       //ancora viene creata
      {
          pM = (Molecola *) CreaMolecola(id_molecola,tipo_molecola);
          //Inserisce la molecola creata nel grafo - vedi grafo.h
          InsMolGrafo(pM,G);
      }
       pA = (Atomo *) CreaAtomo(id_atomo,x,y,z,tipo_atomo);

       //Inserisce l'atomo che è appena stato creato nella molecola
       InsAtomoMolecola(pA, pM);
     }
  }

  fclose(fInputFile);
}

void InsMolGrafo(Molecola *M, Grafo *G)
{
    //Inserisce la molecola in fondo alla lista delle molecole del grafo G
    InsCodaListaMolecole(M,G->LM);

    //Incrementa il contatore
    G->NumMolecole++;
}
```

```
void InsAtomoMolecola(Atomo *A, Molecola *M)
{
    /*ATTENZIONE - prima di effettuare l'inserimento NON viene controllato
    se l'atomo esiste già prima dell'inserimento*/

    /*Se si tratta di idrogeno l'atomo viene aggiunto in coda
    alla lista degli idrogeni*/
    if(A->Elemento[0]=='H')
    {
        InsCodaListaAtomi (A, M->Idrogeni);
        A->M = M;
    }

    /*Se si tratta di AZOTO (N) o OSSIGENO (O) l'atomo viene aggiunto in coda
    alla lista dei donori e accettori

    */
    else if(A->Elemento[0]=='N' || A->Elemento[0]=='O')
    {
        InsCodaListaAtomi (A, M->AccDon);
        A->M = M;
    }

    /*in tutti gli altri casi l'atomo viene aggiunto in coda alla lista degli
    altri atomi*/
    else
    {
        InsCodaListaAtomi (A, M->Altri);
        A->M = M;
    }
}
```

Individuazione dei legami del grafo

```
void StabilisciLegamiGrafo(Grafo *G, float ALFA, float DELTA, char *InputFile)
{
   FILE *fScript; //file contenente lo script per visualizzare gli HB in RasMol
   char nomefScript[80];      //dove 80 è la costante lunghezza definita in
                              //main.c
   Molecola *M, *M1;
   Atomo *H, *D, *A;
   float Dist, Ang;

   sprintf(nomefScript, "%s.scr", InputFile);
   fScript = fopen(nomefScript, "w");
   fprintf(fScript, "set monitor off\n");

   for(M = primoListaMolecole(G->LM); !fineListaMolecole(M,G->LM);
      M=succListaMolecole(M))
      for(H = primoListaAtomi(M->Idrogeni); !fineListaAtomi(H,M->Idrogeni);
         H=succListaAtomi(H))
         for(D = primoListaAtomi(M->AccDon); !fineListaAtomi(D,M->AccDon);
            D=succListaAtomi(D))
              for(M1 = primoListaMolecole(G->LM); !fineListaMolecole(M1,G->LM);
                 M1=succListaMolecole(M1))
                 if (M!=M1)
                    for(A = primoListaAtomi(M1->AccDon);
                         !fineListaAtomi(A,M1->AccDon); A=succListaAtomi(A))
                    {
                      Dist=Distanza(D,A);//calcola distanza donore accettore
                      Ang=Angolo(D,H,A);//Calcola il coseno dell'angolo tra D,H,A
                      if ((Ang<=ALFA) && (Dist<=DELTA))
                               {
                              AggiungiLegame(G,M,M1,D,H,A,Dist,Ang);
                                          fprintf(fScript, "monitor %d %d\n",
                                          H->Indice, A->Indice);
                               }
                     }

      fclose(fScript);
}

float Distanza(Atomo *A1, Atomo *A2) //Calcola distanza tra atomi
{
  float Ris;
  Ris = (float) sqrt(pow((A1->x - A2->x),2) + pow((A1->y - A2->y),2) +
                     pow((A1->z - A2->z),2));
  return Ris;
}

float Angolo(Atomo *D, Atomo *H, Atomo *A) //Calcola il cos tra D H A
{
  float Ris , HD, HA, ProdScal;
    HD = Distanza(D,H);
    HA = Distanza(A,H);
    ProdScal = ((D->x - H->x)*(A->x - H->x) + (D->y - H->y)*(A->y - H->y) +
                (D->z - H->z)*(A->z - H->z));
    Ris = ProdScal/(HD*HA);
  return Ris;
}
```

```
void AggiungiLegame(Grafo *G, Molecola *M, Molecola *M1, Atomo *D,Atomo *H,
                    Atomo *A, float Dist, float Ang)
{
    Legame *L1, *L2;

    L1 = CreaLegame(G->NumLegami+1, Dist, Ang, A, D, H, M, M1);
    L2 = CreaLegame(G->NumLegami+1, Dist, Ang, A, D, H, M1, M);

    L1->Opposto = L2;
    L2->Opposto = L1;

    InsCodaListaLegami(L1, M->Legami);
    InsCodaListaLegami(L2, M1->Legami);

    G->NumLegami++;
    /*printf("E' stato aggiunto il legame %d\n", L1->Indice); */
}
```

Individuazione dei cluster

```
void StabilisciCluster(Grafo *G)
{
 Cluster *C;
 Molecola *M;

 for(M = primoListaMolecole(G->LM); !fineListaMolecole(M,G->LM);
     M=succListaMolecole(M))
 {
         if (M->pC == NULL)
       {
          C=CreaCluster (G->NumCluster + 1);
          InsCodaListaCluster (C, G->LC);
          G->NumCluster++;
          Visita(G, M, C);
       }
 }
}

void Visita(Grafo *G, Molecola *M, Cluster *C)
{
    Legame *pL;
    Molecola *M1;
    pMolecola *pM;

    M->Visitato=TRUE;

    //si inserisce la molecola nel cluster
    pM= CreapMolecola(M);
    InsCodaListapMolecole(pM, C->LPM);
    C->NumMolecole++;

    //si fa puntare la molecola al cluster
    M->pC = C;

    //se si tratta di una molecola d'acqua incrementa il contatore delle
    //molecole d'acqua, altrimenti il contatore soluto
    if(!strcmp(pM->Mol->Nome, "WAT"))
        C->NumAcqua++;
    else
        C->NumSoluto++;

    /*Viene effettuata la scansione delle molecole legate da legami a idrogeno
```

```
    alla attuale, viene verificato se la moleocola è già stata visitata, e in
    caso negativo viene chiamata ricorsivamente la procedura Visita()*/
    for(pL = primoListaLegami(M->Legami); !fineListaLegami(pL,M->Legami);
        pL=succListaLegami(pL))
    {
         M1 = pL->M2;
         if (M1->Visitato == FALSE)
            Visita(G, M1, C);
    }
}
```

Analisi della distribuzione dei cluster

```
void  FrequenzaCluster(Grafo *G, int SOGLIA, char *FrequencyFile)
{
    int *Freq, *Wat, *Ws, *Sol;
    posCluster *pC;
    FILE *fFreq;
    int i;

    Freq = (int*) calloc(SOGLIA+1, sizeof(int));
    Wat = (int*) calloc(SOGLIA+1, sizeof(int));
    Ws = (int*) calloc(SOGLIA+1, sizeof(int));
    Sol = (int*) calloc(SOGLIA+1, sizeof(int));

    for(i=0;i<=SOGLIA;i++)
    {
        Freq[i]=0;
        Wat[i]=0;
        Ws[i]=0;
        Sol[i]=0;
    }

    for(pC = primoListaCluster(G->LC); !fineListaCluster(pC,G->LC);
        pC=succListaCluster(pC))
    {
        if (pC->NumMolecole > SOGLIA)
           Freq[SOGLIA]++;
        else
           Freq[pC->NumMolecole-1]++;

        //Controlla se il cluster ha solo acqua
        if (pC->NumAcqua == pC->NumMolecole)
        {
           if (pC->NumMolecole > SOGLIA)
               Wat[SOGLIA]++;
           else
               Wat[pC->NumMolecole-1]++;
        }

        //Controlla se il cluster ha solo soluto
        else if (pC->NumSoluto == pC->NumMolecole)
        {
           if (pC->NumMolecole > SOGLIA)
               Sol[SOGLIA]++;
           else
               Sol[pC->NumMolecole-1]++;
        }

         //Il cluster ha soluto e acqua
        else
        {
```

```
            if (pC->NumMolecole > SOGLIA)
                Ws[SOGLIA]++;
            else
                Ws[pC->NumMolecole-1]++;
        }
    }

    fFreq=fopen(FrequencyFile,"a+");

    for (i=0;i<=SOGLIA;++i)
        fprintf (fFreq,"%d\t",Freq[i]);

    for (i=0;i<=SOGLIA;++i)
        fprintf (fFreq,"%d\t",Wat[i]);

    for (i=0;i<=SOGLIA;++i)
        fprintf (fFreq,"%d\t",Ws[i]);

    for (i=0;i<=SOGLIA;++i)
        fprintf (fFreq,"%d\t",Sol[i]);

    fprintf(fFreq,"\n");

    fclose(fFreq);
    free(Freq);
}
```

Scrittura del file PDB di output

```
void ScriviPdb(Grafo *G, char *OutputFile)

{
    posCluster *pC;
    pospMolecola *ppM;
    posMolecola *pM;
    posAtomo *pA;
    FILE  *F;            //puntatore al file che contiene i cluster
    int n;
    n=1;

 F = fopen(OutputFile,"w");

 //ciclo per scrivere nel file pdb di output i REMARK che indicano i cluster e
//le molecole che li compongono
 for(pC = primoListaCluster(G->LC); !fineListaCluster(pC,G->LC);
      pC=succListaCluster(pC))
 {
        fprintf(F,"REMARK 4   CLUSTER %d MOL_INIZ %d", pC->Indice, n);

        for(ppM = primoListapMolecole(pC->LPM);
            !fineListapMolecole(ppM,pC->LPM); ppM=succListapMolecole(ppM))
        {
             n++;
        }

        fprintf(F," MOL_FINALE %d\n", n-1);

  }

 n = 1; //ripristina il contatore n a 1 per usarlo per cambiare l'indice delle
        //molecole
```

```
 //ciclo per scrivere l'elenco delle molecole ordinate secondo il cluster nel
file pdb di output
 for(pC = primoListaCluster(G->LC); !fineListaCluster(pC,G->LC);
      pC=succListaCluster(pC))
 {

        for(ppM = primoListapMolecole(pC->LPM);
            !fineListapMolecole(ppM,pC->LPM); ppM=succListapMolecole(ppM))

        {
            pM = ppM->Mol;

            for(pA = primoListaAtomi(pM->AccDon);
                !fineListaAtomi(pA,pM->AccDon); pA=succListaAtomi(pA))
                fprintf(F,"ATOM  %5d  %-3s %-4s %4d    %8.3f%8.3f%8.3f %d\n",
                  pA->Indice, pA->Elemento, pM->Nome, n, pA->x, pA->y, pA->z,
                  pM->Indice);
            for(pA = primoListaAtomi(pM->Idrogeni);
                  !fineListaAtomi(pA,pM->Idrogeni); pA=succListaAtomi(pA))
                fprintf(F,"ATOM  %5d  %-3s %-4s %4d    %8.3f%8.3f%8.3f %d\n",
                  pA->Indice, pA->Elemento, pM->Nome, n, pA->x, pA->y, pA->z,
                  pM->Indice);
            for(pA = primoListaAtomi(pM->Altri); !fineListaAtomi(pA,pM->Altri);
                pA=succListaAtomi(pA))
                fprintf(F,"ATOM  %5d  %-3s %-4s %4d    %8.3f%8.3f%8.3f %d\n",
                  pA->Indice, pA->Elemento, pM->Nome, n, pA->x, pA->y, pA->z,
                  pM->Indice);
            n++;
         }

  }

  fclose(F);

}
```

Funzione di debug per la stampa del grafo in formato leggibile

```
void StampaGrafo(Grafo *G, char *OutputFile)
{
    posMolecola *pM; //puntatore alla posizione della molecola corrente
    posAtomo *pA; //puntatore alla posizione dell'atomo corrente
    posLegame *pL;
    posCluster *pC;
    pospMolecola *ppM;
    FILE *fOutFile;

    fOutFile = fopen(OutputFile,"w");

    printf("Attendi, sto scrivendo su file le molecole\n");

    fprintf(fOutFile,"Il grafo è composto da %d molecole\n",G->NumMolecole);
    fprintf(fOutFile,"Il grafo è composto da %d legami ad idrogeno\n",
            G->NumLegami);
    fprintf(fOutFile,"Il grafo è composto da %d cluster\n",G->NumCluster);


    //effettua la scansione della lista delle molecole
    for(pM = primoListaMolecole(G->LM); !fineListaMolecole(pM,G->LM);
      pM=succListaMolecole(pM))
    {
        fprintf(fOutFile,"Molecola %d: %s\n", pM->Indice, pM->Nome);
```

```
        fprintf(fOutFile,"Questa molecola appartiene al cluster %d\n",
                  pM->pC->Indice);

        //per ogni molecola stampa gli atomi che la compongono scorrendo le
        //varie lista
        for(pA = primoListaAtomi(pM->Idrogeni);
            !fineListaAtomi(pA,pM->Idrogeni); pA=succListaAtomi(pA))
                fprintf(fOutFile,"\tAtomo %d: %.2s di coordinate %.3f %.3f
                        %.3f\n", pA->Indice, pA->Elemento, pA->x, pA->y, pA->z);

        for(pA = primoListaAtomi(pM->AccDon); !fineListaAtomi(pA,pM->AccDon);
            pA=succListaAtomi(pA))
                fprintf(fOutFile,"\tAtomo %d: %.2s di coordinate %.3f %.3f
                        %.3f\n", pA->Indice, pA->Elemento, pA->x, pA->y, pA->z);

        for(pA = primoListaAtomi(pM->Altri); !fineListaAtomi(pA,pM->Altri);
            pA=succListaAtomi(pA))
                fprintf(fOutFile,"\tAtomo %d: %.2s di coordinate %.3f %.3f
                  %.3f\n", pA->Indice, pA->Elemento, pA->x, pA->y, pA->z);

        for(pL = primoListaLegami(pM->Legami); !fineListaLegami(pL,pM->Legami);
            pL=succListaLegami(pL))
                fprintf(fOutFile,"\t\tLegame %d:  dist: %f  angolo: %f  donore:
                        %d (%.2s)  accettore: %d (%.2s) idrogeno: %d (%.2s)\n",
                        pL->Indice, pL->Distanza, pL->Angolo, pL->Don->Indice,
                        pL->Don->Elemento, pL->Acc->Indice, pL->Acc->Elemento,
                        pL->H->Indice, pL->H->Elemento);
    }

    printf("Attendi, sto scrivendo su file i cluster\n");
    fprintf(fOutFile,"\n\n CLUSTER \n");
    for(pC = primoListaCluster(G->LC); !fineListaCluster(pC,G->LC);
        pC=succListaCluster(pC))
    {
        fprintf(fOutFile,"Cluster: %d\n",pC->Indice);
        fprintf(fOutFile, "Il cluster è composto da %d molecole\n",
                  pC->NumMolecole);
        fprintf(fOutFile, "di cui %d molecole d'acqua e %d molecole di
                soluto\n",pC->NumAcqua, pC->NumSoluto);
        for(ppM = primoListapMolecole(pC->LPM);
            !fineListapMolecole(ppM,pC->LPM); ppM=succListapMolecole(ppM))
            fprintf(fOutFile,"\tMolecola %d %s\n",ppM->Mol->Indice,
                        ppM->Mol->Nome);
    }

    fclose(fOutFile);
}
```

Deallocazione delle strutture dati utilizzate

```
void DistruggiGrafo(Grafo *G)
{
   posMolecola *pM;
   posCluster *pC;

  //Distrugge la lista delle molecole
  for (pM = primoListaMolecole(G->LM); !fineListaMolecole(pM,G->LM);
      pM = succListaMolecole(pM))
  {
    distruggeListaLegami(&pM->Legami);
    distruggeListaAtomi(&pM->AccDon);
    distruggeListaAtomi(&pM->Idrogeni);
```

```
    distruggeListaAtomi(&pM->Altri);
  }
  distruggeListaMolecole(&G->LM);

  //Distrugge la lista dei cluster
  for (pC = primoListaCluster(G->LC); !fineListaCluster(pC,G->LC);
      pC = succListaCluster(pC))
  {
    distruggeListapMolecole(&pC->LPM);
  }
  distruggeListaCluster(&G->LC);

  G->NumLegami = 0;
  G->NumMolecole = 0;
  G->NumCluster = 0;

  free(G);
  //printf("grafo distrutto");
}
```

Esempio di implementazione di una lista

Di seguito sono riportate le funzioni di implementazione della lista delle molecole. Le altre liste sono state implementate in modo simile.

```
Molecola *CreaMolecola (int Indice, char *Nome)
{
  Molecola *M = (Molecola *) malloc(sizeof(Molecola));
  if (M == NULL)
  {
    printf("Memoria insufficiente per allocare la Molecola %d!\n",Indice);
    exit(EXIT_MEMORY);
  }

  M->Indice = Indice;
  strcpy(M->Nome, Nome);
  M->Legami = CreaListaLegami(); //Lista dei legami H che interessano la
                                 //molecola
  M->AccDon = CreaListaAtomi(); //Lista degli atomi donori/accettori che
                                //compongono la molecola
  M->Idrogeni = CreaListaAtomi(); //Lista degli atomi di idrogeno che compongono
                                  //la molecola
  M->Altri= CreaListaAtomi(); //Lista degli altri atomi che compongono la
                              //molecola
  M->Succ = M->Prec = NULL;
  M->pC=NULL;
  M->Visitato=FALSE;
  return M;
}

ListaMolecole *CreaListaMolecole () {
  ListaMolecole *L = (ListaMolecole *) malloc(sizeof(ListaMolecole));
  if (L == NULL)
  {
    printf("Memoria insufficiente per allocare la lista delle molecole!\n");
    exit(EXIT_MEMORY);
  }
  L->Succ = L;
  L->Prec = L;
  return L;
}
```

```
void distruggeListaMolecole (ListaMolecole **pL) {
  posMolecola *p;
  for (p = primoListaMolecole(*pL); !ListaMolecoleVuota(*pL);
      cancListaMolecole(&p));
  *pL = NULL;
}

posMolecola *primoListaMolecole (ListaMolecole *L) {
  posMolecola *p = L->Succ;
  return p;
}

posMolecola *ultimoListaMolecole (ListaMolecole *L) {
  posMolecola *p = L->Prec;
  return p;
}

boolean fineListaMolecole (posMolecola *p, ListaMolecole *L) {
  boolean b = (p == L);
  return b;
}

boolean ListaMolecoleVuota (ListaMolecole *L) {
  boolean b = (L->Succ == L) && (L->Prec == L);
  return b;
}

posMolecola *succListaMolecole (posMolecola *p) {
  posMolecola *M = p->Succ;
  return M;
}

void InsMolecola (Molecola *M, posMolecola *p) {
  M->Prec = p->Prec;
  M->Succ = p;
  p->Prec->Succ = M;
  p->Prec = M;
}

void cancListaMolecole (posMolecola **pp) {
  posMolecola *M;
  M = *pp;
  M->Prec->Succ = M->Succ;
  M->Succ->Prec = M->Prec;
  *pp = M->Succ;
  free(M);
}

void InsCodaListaMolecole (Molecola *M, ListaMolecole *L) {
    M->Prec=L->Prec;
    L->Prec->Succ=M;
    L->Prec=M;
    M->Succ=L;
}
```

Analisi del decadimento degli HBs in funzione del tempo

Di seguito è riportata la funzione main() e le funzioni che differiscono da quelle mostrate in precedenza.

```
int main(int argc, char *argv[])
{
  char NomeFileInput[LUNGHEZZA]; //in questa stringa viene memorizzato il nome
                                 //dei file di input
  char InputFile[LUNGHEZZA];
  char NomeOutputFile[LUNGHEZZA];
  char i_stringa[3];
  float ALFA, DELTA;
  int SOGLIA, NumFile, i;
  int *DecadimentoLegami;
  FILE *fDecadimento, *fOutputLegami;
  boolean EsisteLegame=FALSE;
  posMolecola *pM;
  posLegame *pL;

  Grafo **G;

  //Legge il file di configurazione
  LeggeConfigurazione(&ALFA, &DELTA, &SOGLIA);

  // Interpretazione della linea di comando
  LeggeInput(argc,argv, NomeFileInput, &NumFile, NomeOutputFile);

  //Crea dinamicamente un array di puntatori a grafo
  G = (Grafo**) calloc(NumFile+1,sizeof(Grafo*));
  if (G == NULL)
  {
      printf("Memoria insufficiente per allocare il vettore dei puntatori a
            grafi");
    exit(EXIT_MEMORY);
  }

  //Crea dinamicamente un array di interi
  DecadimentoLegami = (int *) calloc(NumFile+1, sizeof(int));
  if (G == NULL)
  {
      printf("Memoria insufficiente per allocare il vettore degli interi per il
            decadimento dei legami");
    exit(EXIT_MEMORY);
  }

  //Inizializza l'array decadimento legami a zero
  for(i=1; i<NumFile+1; i++)
        DecadimentoLegami[i]=0;

  // Crea array di puntatori a grafo
  for (i=1; i<=NumFile; i++)
     G[i]=CreaGrafo();

  // Carica i dati di ogni atomo dei file di input nomefile.i.pdb nel grafo G[i]
  for (i=1; i<=NumFile; i++)
  {
      strcpy(InputFile, NomeFileInput);
      strcat(InputFile, ".");
      sprintf(i_stringa,"%d",i);
      strcat(InputFile, i_stringa);
```

```
    strcat(InputFile, ".pdb");
    printf("Attendi, carico gli atomi del file %s\n", InputFile);
    CaricaAtomi(InputFile,G[i]);
}

// Stabilisce i legami del grafo
for (i=1; i<=NumFile; i++)
{
  printf("Attendi, sto creando i legami per il grafo %d\n", i);
  StabilisciLegamiGrafo(G[i],ALFA,DELTA);
}

//Memorizza il numero dei legami del primo grafo in DecadimentoLegami[1]
DecadimentoLegami[1] = G[1]->NumLegami;

//Questo ciclo stabilisce quali legami si conservano nel tempo (in tutti i
//frames)
fOutputLegami = fopen("legami.txt", "w");
for(pM = primoListaMolecole(G[1]->LM); !fineListaMolecole(pM,G[1]->LM);
    pM=succListaMolecole(pM))
   for(pL = primoListaLegami(pM->Legami); !fineListaLegami(pL,pM->Legami);
         pL=succListaLegami(pL))
   {
        i=2;
        EsisteLegame=TRUE;
        while(i<=NumFile && EsisteLegame==TRUE)
        {
             EsisteLegame = FindLegame(G[i], pL->Acc->Indice, pL->Don->Indice,
                                       pL->H->Indice);

                  if(EsisteLegame == TRUE)
                        DecadimentoLegami[i]++;

                  i++;
        }

        if(i>NumFile && EsisteLegame==TRUE)
             fprintf(fOutputLegami, "donore: %d (%s) mol: %d,\taccettore: %d
                      (%s) mol: %d,\t idrogeno: %d (%s) mol: %d\n",
                       pL->Don->Indice, pL->Don->Elemento, pL->Don->M->Indice,
                       pL->Acc->Indice, pL->Acc->Elemento, pL->Acc->M->Indice,
                       pL->H->Indice, pL->H->Elemento, pL->H->M->Indice);
   }

 fclose(fOutputLegami);

 //Questo ciclo scrive nel file di output le informazioni sul decadimento dei
 //legami
 fDecadimento = fopen(NomeOutputFile, "w");

 for(i=1; i<NumFile+1; i++)
 {
      if (i==1)
          fprintf(fDecadimento, "%d\n", DecadimentoLegami[i]);
      else
          //diviso due perché vengono contati anche i legami opposti
          fprintf(fDecadimento, "%d\n", DecadimentoLegami[i]/2);
 }

 fclose(fDecadimento);
```

```
//Distrugge tutti i grafi
  for (i=1; i<=NumFile; i++)
  {
    DistruggiGrafo(G[i]);
  }

  free(G);

  system("PAUSE");
  return 0;
}

boolean FindLegame(Grafo *G, int Acc, int Don, int H)
{
  posMolecola *pM;
  posLegame *pL;
  boolean Risultato=FALSE;

  for(pM = primoListaMolecole(G->LM); !fineListaMolecole(pM,G->LM);
      pM=succListaMolecole(pM))
     for(pL = primoListaLegami(pM->Legami); !fineListaLegami(pL,pM->Legami);
         pL=succListaLegami(pL))
      if ((pL->Acc->Indice==Acc)&&(pL->Don->Indice==Don)&&(pL->H->Indice==H))
         Risultato = TRUE;
  return Risultato;
}
```

Analisi del decadimento degli HBs in funzione del tempo in prima shell

Rispetto al main descritto nel paragrafo precedente è variata la seguente sezione di codice

```
// Stabilisce i legami del grafo
for (i=1; i<=NumFile; i++)
{
  printf("Attendi, sto creando i legami per il grafo %d\n", i);
  //Stabilisce i legami solo per le molecole presenti in prima shell
    StabilisciLegamiGrafoPrimaShell(G[i],ALFA,DELTA);
}
```

Dove la funzione StabilisciLegamiGrafoPrimaShell() è cosi definita:

```
//Stabilisce i legami a idrogeno delle molecole d'acqua in prima shell
void StabilisciLegamiGrafoPrimaShell(Grafo *G, float ALFA, float DELTA)
{
   Molecola *M, *M1;
   Atomo *H, *D, *A;
   float Dist, Ang;
   for(M = primoListaMolecole(G->LM); !fineListaMolecole(M,G->LM);
       M=succListaMolecole(M))
   {
       //se la molecola è in prima shell
       if(MolecolaPrimaShell(M, G))
       {
                for(H = primoListaAtomi(M->Idrogeni);
                   !fineListaAtomi(H,M->Idrogeni); H=succListaAtomi(H))
                      for(D = primoListaAtomi(M->AccDon);
                         !fineListaAtomi(D,M->AccDon); D=succListaAtomi(D))
                            for(M1 = primoListaMolecole(G->LM);
                                !fineListaMolecole(M1,G->LM);
                                M1=succListaMolecole(M1))
                                  if (M!=M1)
                                      for(A = primoListaAtomi(M1->AccDon);
                                          !fineListaAtomi(A,M1->AccDon);
                                          A=succListaAtomi(A))
                                      {
                                        //calcola distanza donore accettore
                                        Dist=Distanza(D,A);
                                        //calcola l'angolo tra D,H,A
                                        Ang=Angolo(D,H,A);
                                        if ((Ang<=ALFA) && (Dist<=DELTA))
                                        AggiungiLegame(G,M,M1,D,H,A,Dist,Ang);
                                      }
       }
   }
}
```

Distribuzione molecole di acqua in funzione del numero di HBs

Di seguito è riportata la funzione main() modificata per ottenere la distribuzione molecole di acqua in funzione del numero di HBs.

```
int main(int argc, char *argv[])
{
  char NomeFileInput[LUNGHEZZA]; //in questa stringa viene memorizzato il nome
                                 //dei file di input
  char InputFile[LUNGHEZZA];
  char NomeOutputFile[LUNGHEZZA];
  char i_stringa[3];
  float ALFA, DELTA;
  int SOGLIA, NumFile, i, j, temp;
  int **DistLeg;
  FILE *fDistLeg;
  //FILE *fOutputLegami;
  boolean EsisteLegame=FALSE;
  //posMolecola *pM;
  //posLegame *pL;
  Molecola *M;


  Grafo **G;

  //Legge il file di configurazione
  LeggeConfigurazione(&ALFA, &DELTA, &SOGLIA);

  // Interpretazione della linea di comando
  LeggeInput(argc,argv, NomeFileInput, &NumFile, NomeOutputFile);

  //Crea dinamicamente un array di puntatori a grafo
  G = (Grafo**) calloc(NumFile+1,sizeof(Grafo*));
  if (G == NULL)
  {
      printf("Memoria insufficiente per allocare il vettore dei puntatori a
            grafi");
    exit(EXIT_MEMORY);
  }

  //Crea dinamicamente un array bidimensionale di interi
  DistLeg = (int **) calloc(NumFile,sizeof(int *));
      for(i=1; i<NumFile+1; i++)
      {
            DistLeg[i] = (int *) calloc(6,sizeof(int));
      }
  if (G == NULL)
  {
      printf("Memoria insufficiente per allocare la matrice per la distribuzione
            dei legami");
    exit(EXIT_MEMORY);
  }

  //Inizializza l'array della distribuzione dei legami
  for(i=1; i<NumFile+1; i++)
        for(j=0; j<6; j++)
            DistLeg[i][j]=0;

  // Crea array di puntatori a grafo
  for (i=1; i<NumFile+1; i++)
     G[i]=CreaGrafo();
```

```
  // Carica i dati di ogni atomo dei file di input nomefile.i.pdb nel grafo G[i]
  for (i=1; i<NumFile+1; i++)
  {
      strcpy(InputFile, NomeFileInput);
      strcat(InputFile, ".");
      sprintf(i_stringa,"%d",i);
      strcat(InputFile, i_stringa);
      strcat(InputFile, ".pdb");
      printf("Attendi, carico gli atomi del file %s\n", InputFile);
      CaricaAtomi(InputFile,G[i]);
  }

  // Stabilisce i legami del grafo
  for (i=1; i<NumFile+1; i++)
  {
    printf("Attendi, sto creando i legami per il grafo %d\n", i);
    StabilisciLegamiGrafo(G[i],ALFA,DELTA);
  }

  //Conteggia la distribuzione del numero degli HB tra le molecole d'acqua
  for (i=1; i<NumFile+1; i++)
  {
      for(M = primoListaMolecole(G[i]->LM); !fineListaMolecole(M,G[i]->LM);
          M=succListaMolecole(M))
      {
        if(!strcmp(M->Nome,"WAT"))
        {
             temp = M->NumHB;
               (DistLeg[i][temp])++;
        }
      }
  }


  fDistLeg = fopen(NomeOutputFile, "w");
   for(i=1; i<NumFile+1; i++)
   {
         for(j=0; j<6; j++)
         {
                  fprintf(fDistLeg, "%d ", DistLeg[i][j]);
         }

         fprintf(fDistLeg, "\n");
   }
   fclose(fDistLeg);


  //Distrugge tutti i grafi
  for (i=1; i<NumFile+1; i++)
  {
    DistruggiGrafo(G[i]);
  }

  free(G);

  system("PAUSE");
  return 0;
}
```

Distribuzione molecole di acqua in funzione del numero di HBs in prima shell

Di seguito è riportata la funzione main() modificata per ottenere la distribuzione molecole di acqua in funzione del numero di HBs in prima shell.

```
int main(int argc, char *argv[])
{
  char NomeFileInput[LUNGHEZZA]; //in questa stringa viene memorizzato il nome
                                 //dei file di input
  char InputFile[LUNGHEZZA];
  char NomeOutputFile[LUNGHEZZA];
  char i_stringa[3];
  float ALFA, DELTA;
  int SOGLIA, NumFile, i, j, temp;
  int NumMolPrimaShell=0; //numero delle molecole d'acqua in prima shell
  int **DistLeg;
  FILE *fDistLeg;
  //FILE *fOutputLegami;
  boolean EsisteLegame=FALSE;
  Molecola *Sol;  //puntatore alla molecola di soluto
  //posLegame *pL;
  Molecola *M;
  Atomo *A, *O; //puntatore generico ad atomo e puntatore all'atomo di ossigeno
                //dell'acqua
  float DistMin = 10000; //Valore in cui si memorizza la distanza minima tra
                         //l'ossigeno dell'acqua e gli atomi del soluto
  float Dist = 0; //Usata per memorizzare la distanza tra atomi

  Grafo **G;

  //Legge il file di configurazione
  LeggeConfigurazione(&ALFA, &DELTA, &SOGLIA);

  // Interpretazione della linea di comando
  LeggeInput(argc,argv, NomeFileInput, &NumFile, NomeOutputFile);

  //Crea dinamicamente un array di puntatori a grafo
  G = (Grafo**) calloc(NumFile+1,sizeof(Grafo*));
  if (G == NULL)
  {
      printf("Memoria insufficiente per allocare il vettore dei puntatori a
            grafi");
    exit(EXIT_MEMORY);
  }

  //Crea dinamicamente un array bidimensionale di interi
  DistLeg = (int **) calloc(NumFile,sizeof(int *));
      for(i=1; i<NumFile+1; i++)
      {
            DistLeg[i] = (int *) calloc(6,sizeof(int));
      }
  if (G == NULL)
  {
      printf("Memoria insufficiente per allocare la matrice per la distribuzione
            dei legami");
    exit(EXIT_MEMORY);
  }

  //Inizializza l'array della distribuzione dei legami
  for(i=1; i<NumFile+1; i++)
```

```
      for(j=0; j<6; j++)
          DistLeg[i][j]=0;

// Crea array di puntatori a grafo
for (i=1; i<NumFile+1; i++)
   G[i]=CreaGrafo();

// Carica i dati di ogni atomo dei file di input nomefile.i.pdb nel grafo G[i]
for (i=1; i<NumFile+1; i++)
{
    strcpy(InputFile, NomeFileInput);
    strcat(InputFile, ".");
    sprintf(i_stringa,"%d",i);
    strcat(InputFile, i_stringa);
    strcat(InputFile, ".pdb");
    printf("Attendi, carico gli atomi del file %s\n", InputFile);
    CaricaAtomi(InputFile,G[i]);
}

// Stabilisce i legami del grafo
for (i=1; i<NumFile+1; i++)
{
  printf("Attendi, sto creando i legami per il grafo %d\n", i);
  StabilisciLegamiGrafo(G[i],ALFA,DELTA);
}

//Conteggia la distribuzione del numero degli HB tra le molecole d'acqua in
//prima shell
//NOTA: Adatto solo con UNA molecola di soluto
for (i=1; i<NumFile+1; i++)
{
    //Ciclo per individuare la molecola di soluto
    for(M = primoListaMolecole(G[i]->LM); !fineListaMolecole(M,G[i]->LM);
        M=succListaMolecole(M))
    {
      if(strcmp(M->Nome,"WAT"))   //se non è una molecola d'acqua
      {
           Sol = M;
             break;
      }
    }

    //Ciclo per la scansione delle molecole d'acqua
    for(M = primoListaMolecole(G[i]->LM); !fineListaMolecole(M,G[i]->LM);
        M=succListaMolecole(M))
    {
      if(!strcmp(M->Nome,"WAT"))
      {
             //Scorre gli atomi della molecola di acqua alla ricerca
             //dell'ossigeno
             for(A = primoListaAtomi(M->AccDon); !fineListaAtomi(A,M->AccDon);
                A=succListaAtomi(A))
             {
                   if(!strcmp(A->Elemento,"O"))     //se l'atomo è di ossigeno
                      {
                            O = A;
                      break;
                      }
             }

             //Scorre le liste degli atomi accettori-donori del soluto
          for(A = primoListaAtomi(Sol->AccDon); !fineListaAtomi(A,Sol->AccDon);
             A=succListaAtomi(A))
```

```
        {
            Dist=Distanza(O,A); //calcola distanza
            //se la distanza è inferiore di DistMin, aggiorna DistMin
               if (Dist < DistMin)
                     DistMin = Dist;
        }

            //Scorre le liste degli atomi idrogeni del soluto
        for(A = primoListaAtomi(Sol->Idrogeni);
            !fineListaAtomi(A,Sol->Idrogeni); A=succListaAtomi(A))
        {
             Dist=Distanza(O,A);//calcola distanza
             //se la distanza è inferiore di DistMin, aggiorna DistMin
                     if (Dist < DistMin)
                           DistMin = Dist;
        }

             //Scorre le liste degli altri atomi del soluto
        for(A = primoListaAtomi(Sol->Altri); !fineListaAtomi(A,Sol->Altri);
            A=succListaAtomi(A))
        {
               Dist=Distanza(O,A);//calcola distanza
             //se la distanza è inferiore di DistMin, aggiorna DistMin
                     if (Dist < DistMin)
                           DistMin = Dist;
        }


         //A questo punto DistMin contiene la distanza minima tra l'ossigeno
         //della molecole d'acqua e gli atomi del soluto, se tale distanza è
         //inferiore a 3.5 l'acqua è in prima shell, viene incrementato il
         //contatore di uno e vengono conteggiati i legami i legami HB di
         //quella molecola
            if(DistMin <= 3.5)
            {
                     NumMolPrimaShell++;
                     temp = M->NumHB;
                     (DistLeg[i][temp])++;
            }
      }

      DistMin = 10000;

    }
}


//Questo ciclo scrive nel file di output le informazioni sulla distribuzione
//degli HB tra le molecole d'acqua in prima shell
 fDistLeg = fopen(NomeOutputFile, "w");
 fprintf(fDistLeg, "Numero medio di molecole in prima shell: %d\n",
          NumMolPrimaShell/NumFile);
 for(i=1; i<NumFile+1; i++)
 {
       for(j=0; j<6; j++)
       {
                fprintf(fDistLeg, "%d ", DistLeg[i][j]);
       }

       fprintf(fDistLeg, "\n");
 }
 fclose(fDistLeg);
```

```
  //Distrugge tutti i grafi
  for (i=1; i<NumFile+1; i++)
  {
    DistruggiGrafo(G[i]);
  }

  free(G);

  system("PAUSE");
  return 0;
}
```

BIBLIOGRAFIA

[1] http://it.wikipedia.org/wiki/Metanolo.

[2] Andrea Bazzoli, Valeria Tacca. Simulazione di dinamica molecolare di una molecola di metanolo in acqua TIP3P. *Note del polo, 5 (dicembre 2002), Dipartimento di Tecnologie dell'Informazione, Università degli studi di Milano.*

[3] http://www.rcsb.org/pdb.

[4] W.L. Jorgensen, J. Chandrasekhar, J.D. Madura, R.W. Impey, M.L. Klein, Comparison of simple potential function for simulating liquid water. *J. Chem. Phys. 79(1983) 926.*

[5] http://it.wikipedia.org/wiki/Urea.

[6] Barbara Bonomi, Ruggero Rizzo. Simulazione di dinamica molecolare di una molecola di urea in acqua. *Corso di Bioinformatica A.A. 2001/2002, Università degli studi di Milano, Dipartimento di Tecnologie dell'Informazione (dicembre 2002).*

[7] Maurizio Sironi, Arianna Fornili, Sandro L. Fornili, Water interaction with glycine betaine: A hybrid QM/MM molecular dynamics simulation, *Phys. Chem. Chem. Phys. 3 (2001) 1081*.

[8] http://it.wikipedia.org/wiki/Prolina.

[9] Arianna Fornili, Monica Civera, Maurizio Sironi, Sandro L. Fornili, Molecular dynamic simulation of acqueous solutions of trimethylamine-N-oxide and tert-butyl-alcohol. *Phys. Chem. Chem. Phys. 5 (2003) 4905.*

[10] Luca Trampetti, Giuseppe Onori, Fernando Pirani, Studio dell'idratazione e aggregazione di gruppi apolari mediante spettrofotometria IR. *Tesi di laurea A.A. 2002-2003, Corso di Laurea in Fisica, Università degli studi di Perugia*.

[11] H. Xu, B.J. Berne, *J. Phys. Chem. B 105 (2001) 386.*