Università degli studi di Udine, AA 2012/2013

**Progetto di Ricerca Operativa**

# Alberi filogenetici minimali consistenti con un insieme di triple radicate

Prezza Nicola
prezza.nicola@spes.uniud.it

6 maggio 2013

# Indice

# 1 Introduzione

La filogenetica è la disciplina che studia l'evoluzione di gruppi di organismi. Dato un insieme di specie e una serie di informazioni quali la composizione genetica degli individui che vi fanno parte, l'obiettivo di questa disciplina è solitamente quello di produrre un *albero filogenetico* dell'insieme che associa ad ogni foglia una specie e ad ogni nodo interno un'evento di speciazione. Il primo passo per raggiungere questo obiettivo è la costruzione di un albero filogenetico relativo ad una famiglia di geni $\mathcal{G}$ proveniente dall'insieme di specie $\mathcal{S}$; in particolare, ogni elemento di $\mathcal{S}$ può essere identificato da un sottoinsieme di $\mathcal{G}$ disgiunto dagli altri elementi in $\mathcal{S}$. Gli eventi che interessano i geni includono (oltre alla speciazione) anche la duplicazione, ossia una biforcazione della linea evolutiva *all'interno* della stessa specie. Essendoci una relazione molto stretta tra $\mathcal{G}$ ed $\mathcal{S}$, è chiaro che i relativi alberi filogenetici devono essere compatibili. Il problema di rendere compatibili le due strutture prende il nome di *riconciliazione degli alberi filogenetici* (si veda figura 1): l'obiettivo è riconciliare i due alberi in modo consistente, ossia facendo corrispondere ad ogni speciazione genica una speciazione tra specie, mantenendo le duplicazioni all'interno della stessa linea evolutiva e rispettando infine la relazione tra $\mathcal{G}$ ed $\mathcal{S}$.

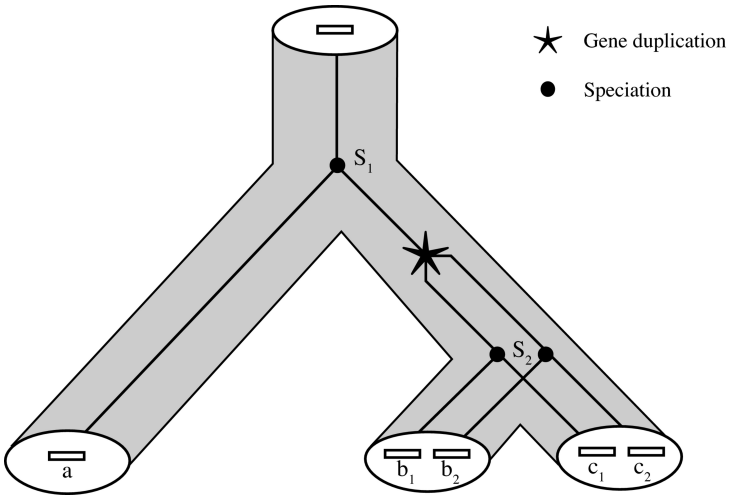


Figura 1: Esempio di riconciliazione degli alberi filogenetici dell'insieme di specie $\{a, b, c\}$ e dell'insieme di geni $\{a, b_1, b_2, c_1, c_2\}$

Il problema della riconciliazione degli alberi filogenetici di specie/geni può essere affrontato anche in un altro modo, ossia partendo solo dall'informazione riguardante le relazioni tra i geni: la cosiddetta relazione di ortologia.

Due geni sono detti tra loro ortologhi se il loro lowest common ancestor è un evento di speciazione, altrimenti sono detti paraloghi. Come mostrato in [7], la relazione di ortologia di una famiglia di geni può essere *approssimata* senza la conoscenza del corrispondente albero filogenetico; a partire dalla relazione di ortologia e dall'informazione di appartenenza dei geni alle specie, il problema della ricostruzione e riconciliazione di entrambi gli alberi filogenetici può essere risolto in modo elegante con un unico modello di programmazione lineare intera[8]. L'output di questo processo consiste in un insieme di *triple radicate di specie*. Date tre specie $i, j, k \in \mathcal{S}$, la tripla radicata $ij|k$ rappresenta un vincolo che le tre specie devono rispettare nell'albero filogenetico: devono esistere due vertici distinti $u \neq v$ e i cammini a due a due internamente disgiunti $u \to i$, $u \to j$, $v \to u$ e $v \to k$[10]. Su un albero questo vincolo può essere espresso in modo alternativo col fatto che il LCA tra $i$ e $j$ deve trovarsi più in basso rispetto al LCA tra $i$ e $k$ (si veda figura 2).

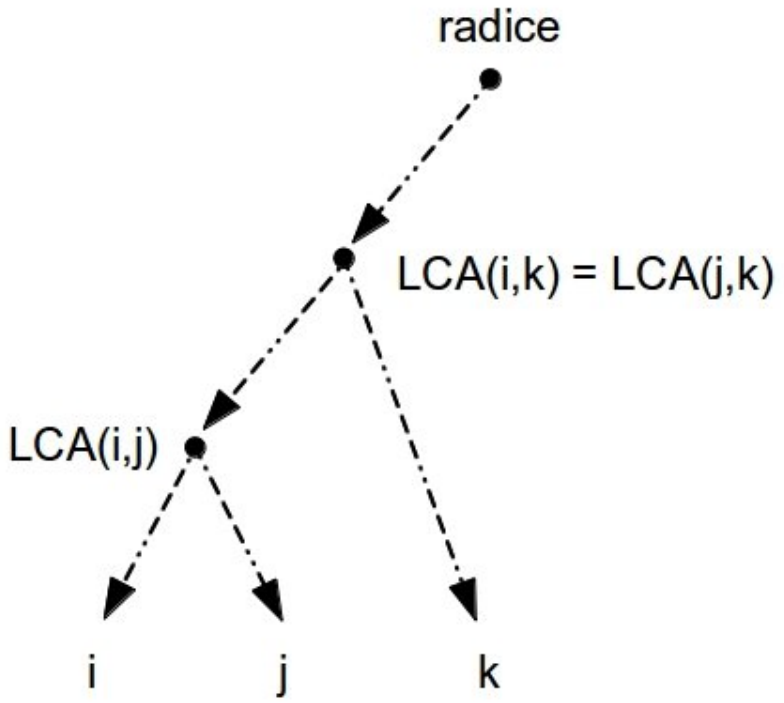


Figura 2: Per rispettare il vincolo $ij|k$ l'albero filogenetico delle specie deve possedere questa topologia. Le linee tratteggiate indicano cammini nell'albero.

Dato un insieme di triple radicate $\mathcal{T}$, possono esistere diversi alberi fi-

logenetici consistenti con esse. Un albero filogenetico consistente con $\mathcal{T}$ è *minimale* se il suo numero di nodi interni è minore o uguale a quello di tutti gli alberi consistenti con $\mathcal{T}$. Per il criterio di massima parsimonia[2], la ricostruzione con il minimo numero di eventi evolutivi corrisponde con maggior probabilità alla realtà. Purtroppo (come spesso accade), il problema di inferire un albero filogenetico minimale consistente con un insieme di triple è NP-hard[6], quindi sono necessarie delle euristiche per risolvere il problema di ottimizzazione. Una delle più note euristiche è dovuta ad Aho et al. e prende il nome di BUILD[4]. Questo algoritmo polinomiale costruisce un albero filogenetico consistente con un insieme di triple, ma non offre garanzie di ottimalità (infatti esistono casi per i quali l'algoritmo restituisce un albero molto più grande dell'ottimo[6]). Un valido tool che implementa diverse euristiche per la risoluzione del problema è Lev1athan[3]: questo programma è stato utilizzato per comparare i risultati generati dal modello descritto di seguito.

In questo progetto si mira a risolvere il problema dell'albero filogenetico minimale consistente con un insieme di triple radicate tramite un modello di programmazione lineare intera mista. Soluzioni basate sulla PLI sono già presenti in letteratura per la risoluzione di problemi correlati[9], ma prendono in input tipi di vincoli diversi (SNPs anzichè triple radicate). L'idea del modello qui presentato è quella di considerare la matrice di adiacenza dell'albero filogenetico, imponendo vincoli affinchè descriva un albero e rispetti l'insieme di triple radicate; la funzione obiettivo da minimizzare è il numero di nodi interni utilizzati.

# 2 Modello MILP

Il modello è articolato in quattro parti principali. Per cominciare (sezione 2.1) occorre mantenere una matrice di adiacenza per descrivere l'albero, imponendo opportuni vincoli affinchè il grafo risultante sia effettivamente un albero (un solo padre per ogni nodo esclusa la radice e un numero di figli maggiore o uguale a 2 per tutti i nodi escluse le foglie). Da questa matrice di adiacenza viene ricavato il numero di nodi interni utilizzati: questo numero va minimizzato quindi corrisponde alla funzione obiettivo. La sezione 2.2 descrive i vincoli e le variabili utilizzate per calcolare la chiusura transitiva dell'albero trasposto (direzione degli archi invertita); questo grafo si rivela necessario per il calcolo dei *lowest common ancestors* dell'albero originale (sezione 2.3), utilizzati per imporre la consistenza dell'albero con l'insieme di triple radicate in input (sezione 2.4).

## 2.1 Matrice di adiacenza

Sia $\mathcal{S}$ l'insieme delle $n$ specie. Dato che solo le foglie dell'albero dovranno essere etichettate con le specie, occorre aggiungere dei nodi interni. Il caso pessimo (numero di nodi interni massimo) è quello di un albero binario, quindi occorre introdurre $n-1$ nodi interni (alcuni dei quali potranno non essere utilizzati). Siano dunque $0, ..., n-2$ i nodi interni e $n-1, ..., 2n-2$ le foglie. Le foglie non possono avere figli, pertanto la matrice di adiacenza $T_{ij}$ è formata da $n-1$ righe corrispondenti ai nodi interni e $2n-1$ colonne. $T_{ij}$ è una matrice binaria (gli archi non sono pesati), quindi il primo vincolo da imporre è

$$T_{ij} \in \{0,1\} \qquad 0 \leq i < n-1,\ 0 \leq j < 2n-1 \tag{1}$$

Dato che in un albero i nodi possono essere ordinati in modo tale che ogni nodo può raggiungere solo nodi successivi nell'ordine (dove la radice è il primo nodo e le foglie sono gli ultimi nodi), si può semplificare il modello introducendo solo variabili $T_{ij}$ tali che $i < j$. Questo vincolo impedisce inoltre la ciclicità del grafo. Senza perdita di generalità, la radice è il nodo con indice 0. In figura 3 è riportato un esempio di numerazione dei nodi di un albero filogenetico. La corrispondente matrice $T_{ij}$ è riportata nella tabella

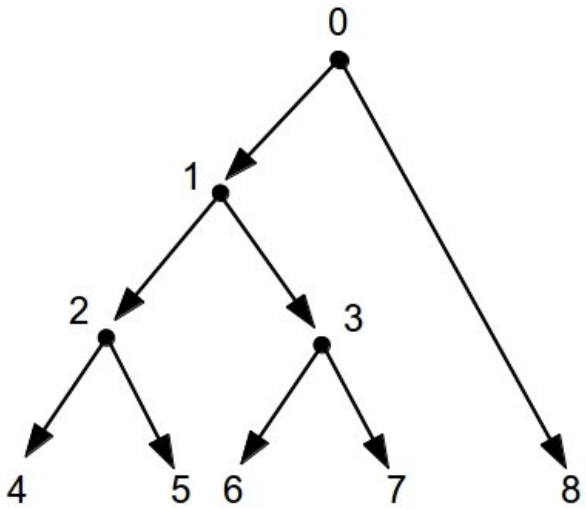


Figura 3: Esempio di albero filogenetico. Le specie sono etichettate con $4, ..., 8$

1. Per diverse ragioni è utile mantenere un vettore binario $u_i$ che per ogni nodo interno mantenga l'informazione se esso è presente o meno nell'albero.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | - | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | - | - | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | - | - | - | 0 | 1 | 1 | 0 | 0 | 0 |
| 3 | - | - | - | - | 0 | 0 | 1 | 1 | 0 |

Tabella 1: Matrice di adiacenza relativa all'albero in figura 3. Le variabili contrassegnate con "-" non sono presenti nel modello.

$u_i$ si può calcolare imponendo che

$$\begin{aligned} & 0 \leq u_i \leq 1 \qquad i = 0, ..., n-2 \\ & u_i \geq T_{ij} \qquad 0 \leq i < n-1,\ i < j < 2n-1 \\ & u_i \leq \sum_{j=i+1}^{2n-2} T_{ij} \quad i = 0, ..., n-2 \end{aligned} \tag{2}$$

Il primo vincolo fissa il dominio di $u_i$, cioè l'intervallo chiuso da 0 a 1; grazie ai vincoli successivi $u_i$ può però assumere solo valori interi. Il secondo vincolo impone che se un nodo interno $i$ ha almeno un figlio allora $u_i$ debba valere 1; in questo caso infatti il nodo è presente nell'albero. Se un nodo interno $i$, al contrario, non ha figli significa che non è presente nell'albero: la somma $\sum_{j=i+1}^{2n-2} T_{ij}$ vale 0 e $u_i$ è forzato a valere 0 grazie al terzo vincolo. Dato che la radice deve essere presente nell'albero, viene imposto

$$u_0 = 1 \tag{3}$$

Per fare sì che la matrice $T_{ij}$ descriva un albero occorre innanzitutto imporre che ogni nodo *utilizzato* esclusa la radice abbia uno e un solo padre. Il vincolo si traduce quindi in

$$\begin{aligned} & \sum_{i=0}^{j-1} T_{ij} = u_j \quad j = 1, ..., n-2 \\ & \sum_{i=0}^{n-2} T_{ij} = 1 \qquad j = n-1, ..., 2n-2 \end{aligned} \tag{4}$$

La prima disuguaglianza riguarda i nodi interni, mentre la seconda le foglie. Un albero è caratterizzato inoltre dal fatto che ogni nodo interno *utilizzato*

ha due o più figli. Questo vincolo si può esprimere nel seguente modo:

$$\sum_{j=i+1}^{2n-2} T_{ij} \geq 2u_i \quad i = 0, ..., n-2 \tag{5}$$

Si noti che questo vincolo rende ridondante il terzo vincolo in (2), che quindi può essere rimosso. Dato che l'obiettivo del modello è minimizzare il numero di nodi interni utilizzati, il vettore $u_i$ si rivela infine fondamentale per la definizione della funzione obiettivo:

$$\textbf{min} \sum_{i=0}^{n-2} u_i \tag{6}$$

Si noti che non occorre imporre che l'albero sia connesso (non sia cioè una foresta) dato che solo un nodo - la radice - può non avere il padre (quindi non possono esistere due alberi sconnessi in quanto si avrebbero due radici senza padre).

## 2.2 Chiusura transitiva dell'albero trasposto

Per motivi descritti di seguito, per il calcolo dei lowest common ancestors in sezione 2.3 occorre calcolare la chiusura transitiva $T_{ij}^C$ di $T_{ij}^T$, ossia della matrice $T_{ij}$ trasposta. Questo obiettivo può essere raggiunto mediante l'introduzione di variabli non intere $P_{ikj},\ 0 \leq j < k < i \leq 2n-2$ tali che $P_{ikj} = 1$ se e solo se in $T_{ij}^T$ esiste un cammino da $i$ a $j$ passante per $k$. I vincoli descritti di seguito implicheranno automaticamente l'interezza delle variabili in $P_{ikj}$ e in $T_{ij}^C$ senza doverla imporre esplicitamente. Innanzitutto occorre fissare limitazioni inferiori e superiori per le variabili:

$$\begin{aligned} &0 \leq T_{ij}^C \leq 1 \quad 0 \leq j < i \leq 2n-2 \\ &0 \leq P_{ikj} \leq 1 \quad 0 \leq j < k < i \leq 2n-2 \end{aligned} \tag{7}$$

Se l'arco da $j$ a $i$ è presente nell'albero originale, allora deve essere presente l'arco da $i$ a $j$ nella chiusura transitiva:

$$T_{ij}^C \geq T_{ji} \quad 0 \leq j \leq n-2,\ j < i \leq 2n-2 \tag{8}$$

Se l'arco da $j$ a $i$ non è presente nell'albero originale e nella chiusura transitiva non esistono cammini da $i$ a $j$ passanti per qualche nodo $k$, nella chiusura transitiva non può esistere l'arco da $i$ a $j$:

$$T_{ij}^C \leq T_{ji} + \sum_{k=j+1}^{i-1} P_{ikj} \quad 0 \leq j < i \leq 2n-2 \tag{9}$$

Viceversa, se esiste almeno un nodo $k$ tale che nella chiusura transitiva esiste il cammino da $i$ a $j$ passante per $k$, allora deve esistere un arco da $i$ a $j$:

$$T_{ij}^C \geq P_{ikj} \quad 0 \leq j < k < i \leq 2n-2 \tag{10}$$

Infine, $P_{ikj} = 1$ se e solo se $T_{ik}^C = 1$ e $T_{kj}^C = 1$:

$$\begin{aligned} 1 + P_{ikj} &\geq T_{ik}^C + T_{kj}^C \quad 0 \leq j < k < i \leq 2n-2 \\ P_{ikj} &\leq T_{ik}^C \qquad\qquad\quad 0 \leq j < k < i \leq 2n-2 \\ P_{ikj} &\leq T_{kj}^C \qquad\qquad\quad 0 \leq j < k < i \leq 2n-2 \end{aligned} \tag{11}$$

Si noti che la disuguaglianza 9 impone che se una variabile $T_{ij}^C$ vale 1 *e* $T_{ji} = 0$ allora deve per forza esistere un nodo intermedio nel cammino da $i$ a $j$. Questo ragionamento si ripete in cascata ai sottocammini intermedi e soddisfa i vincoli solamente se il cammino in ultima analisi è composto da archi presenti nell'albero originale.

## 2.3 Lowest common ancestors

Avendo a disposizione la chiusura transitiva dell'albero trasposto, il calcolo dei LCA può essere eseguito definendo una matrice $L_{ijk}$ dove $i < j$ sono foglie e $k$ è un nodo interno (ossia $0 \leq k < n-1 \leq i < j \leq 2n-2$) tale che $L_{ijk} = 1$ se e solo se nell'albero trasposto esistono i cammini da $i$ a $k$ e da $j$ a $k$:

$$\begin{aligned} 0 &\leq L_{ijk} \leq 1 \qquad\qquad\quad 0 \leq k < n-1 \leq i < j \leq 2n-2 \\ 1 + L_{ijk} &\geq T_{ik}^C + T_{jk}^C \quad 0 \leq k < n-1 \leq i < j \leq 2n-2 \\ L_{ijk} &\leq T_{ik}^C \qquad\qquad\quad 0 \leq k < n-1 \leq i < j \leq 2n-2 \\ L_{ijk} &\leq T_{jk}^C \qquad\qquad\quad 0 \leq k < n-1 \leq i < j \leq 2n-2 \end{aligned} \tag{12}$$

A questo punto si noti che la quantità

$$\sum_{k=0}^{n-2} L_{ijk}$$

è uguale al numero di nodi raggiungibili sia da $i$ che da $j$ durante il cammino verso la radice, quindi è pari all'altezza dell'LCA tra $i$ e $j$ (dove la radice ha altezza 1).

### 2.4 Consistenza con l'insieme di triple

Indicando con $LCA(i,j)$ l'altezza del lowest common ancestor tra $i$ e $j$, per ogni tripla radicata di foglie $ij|k$ deve valere che $LCA(i,j) > LCA(i,k)$ (come mostrato in figura 2). Grazie alle variabili definite nella sezione precedente questo vincolo si può esprimere in modo molto semplice come segue:

$$\sum_{v=0}^{n-2} L'_{ijv} > \sum_{v=0}^{n-2} L'_{ikv} \quad \forall\ ij|k \tag{13}$$

dove $L'_{ijv} = L_{min(i,j)max(i,j)v}$.

## 3 Implementazione

Dato l'elevato numero di vincoli e variabili introdotte (cubico nel numero di specie) si è scelto di implementare il modello utilizzando il risolutore di sistemi lineari (e non) CPLEX, messo a disposizione dalla libreria Java *cplex.jar* (un'implementazione LINGO non sarebbe stata possibile a causa dei limiti imposti dalla versione demo del software). Data la lunghezza del codice (più di 500 linee) esso non viene riportato in questa relazione. Il programma prende in input da linea di comando il percorso del file contenente le triple e un lower bound per il numero di nodi interni dell'albero (compresa la radice) per limitare i tempi di calcolo:

```
java -cp <lib>:<bin> phylTree.Main <triples> <lb>
```

dove $< lib >$ è il percorso contenente la libreria cplex.jar, $< bin >$ è il percorso dell'eseguibile, $< triples >$ è il file contenente le triple e $< lb >$ è il lower bound alla funzione obiettivo (viene aggiunto il vincolo $obj \geq lb$). Il file $< triples >$ deve contenere una tripla ad ogli linea; le triple $AB|C$ devono essere nel formato

```
A B C
```

dove le specie sono separate da uno spazio. Il programma java genera runtime in output informazioni riguardanti la risoluzione del modello e, al termine, la matrice di adiacenza e l'albero corrispondente in formato newick[1]. L'albero filogenetico può essere visualizzato utilizzando tool online come phyfi [5] che stampano l'albero in formato grafico data la stringa newick.

# 4 Test e risultati

Il modello è stato testato su alcune istanze provenienti da un insieme formato da 36 file di triple generate tramite il modello ILP descritto in [8]. I risultati sono stati comparati con gli alberi prodotti dal tool Lev1athan[3]. Dati i lunghi tempi di esecuzione per istanze di grande dimensione, in alcuni casi il modello è stato testato imponendo come lower bound alla funzione obiettivo il numero di nodi interni generati da Lev1athan decrementato di uno (per tentare di migliorare il risultato del tool). Per fare un piccolo esempio, l'istanza *ds_splits05-l.txt* è formata da 7 specie vincolate dalle seguenti triple (22 in totale):

```
D I N     D J N     D K N     D L N     D M N     I J N
I K N     I L N     I M N     J K D     J K I     J K L
J K N     J L D     J L I     J L N     J M N     K L D
K L I     K L N     K M N     L M N
```

data la piccola dimensione dell'istanza, il modello è stato lanciato senza un lower bound per la funzione obiettivo (lb=0). Il newick tree restituito dal programma implementato è il seguente:

```
((((J,K,M),L),D,I),N);
```

I risultati sono mostrati in figura 4; il modello sviluppato ha fornito una soluzione di costo equivalente a quella proposta da Lev1athan (3 nodi interni), ma con una diversa topologia. Su istanze di dimensione maggiore il modello MILP sviluppato ha dimostrato di poter raggiungere risultati migliori di Lev1athan al prezzo di un maggior tempo di esecuzione. L'istanza *ds_splits14-l.txt* è composta da 16 specie e 296 triple; l'albero fornito da Lev1athan (figura 5) è composto da 4 nodi interni, mentre l'albero fornito dal modello MILP (figura 6) da 3:

```
((C1,E1,F1),((A1,S,T,X),O,Q,R,Y,Z),C,E,G,J);
```

Il risolutore CPLEX ha impiegato 375 secondi per trovare la soluzione mostrata. Anche in questo caso si ha una prova di ottimalità in quanto il modello è stato lanciato senza lower bound. Istanze di dimensione maggiore hanno richiesto molto più tempo per essere risolte. Un altro caso nel quale il modello MILP ha prodotto un risultato migliore di Lev1athan è dato dall'istanza *ds_splits15-l.txt*, composta da 17 specie e 390 triple. Lev1athan dopo alcuni secondi propone un albero (figura 7) avente 7 nodi interni, mentre il modello MILP dopo 11 minuti propone un albero (figura 8) avente 6 nodi interni:

```
(((X,Z),Y),(G1,H1),(C1,D,E1,F1),((K,Q,S,U),I,M,N,T));
```

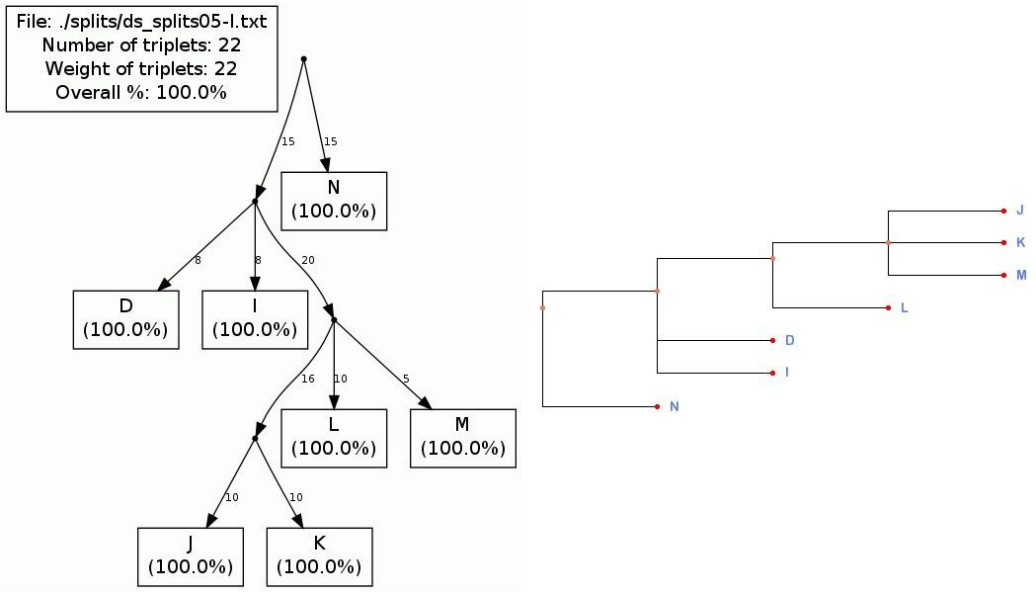


Figura 4: Confronto tra gli alberi prodotti da Lev1athan (sinistra) e dal modello MILP (destra) sul dataset *ds_splits05-l.txt*. I due alberi hanno lo stesso numero di nodi interni (3). Dato che il modello MILP è stato risolto senza limitazione inferiore alla fuzione obiettivo, entrambi gli alberi sono minimali.

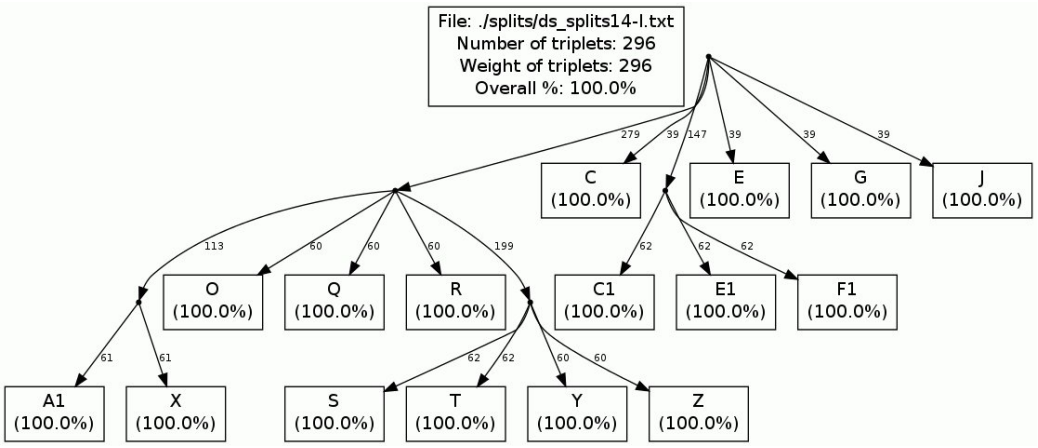


Figura 5: Output di Lev1athan sull'istanza *ds_splits14-l.txt*

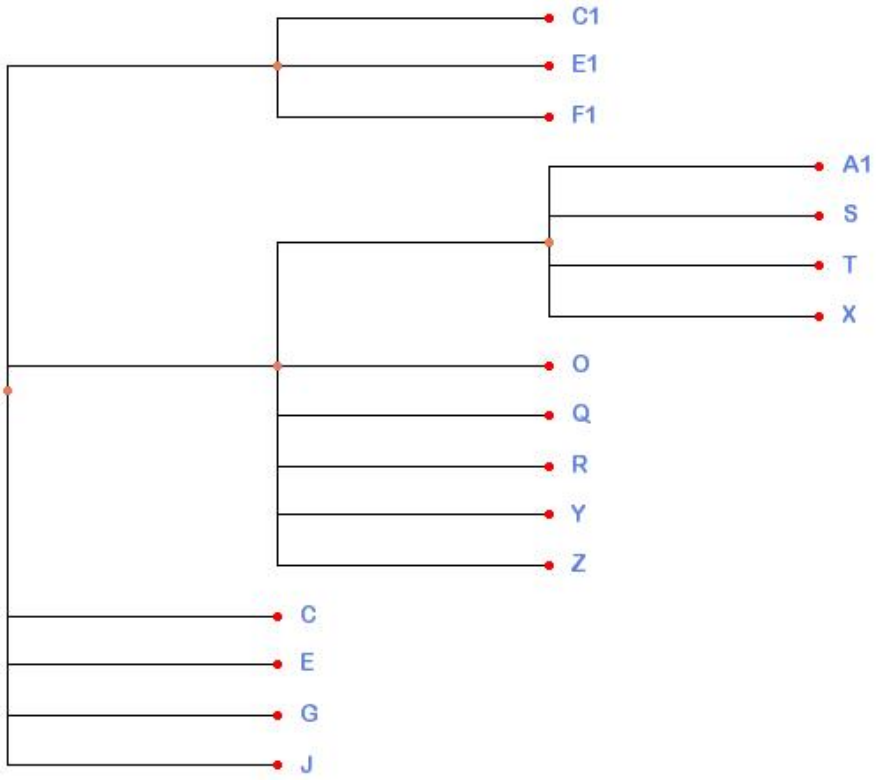


Figura 6: Output del modello MILP sull'istanza *ds_splits14-l.txt*. Questo albero ha meno nodi interni di quello restituito da Lev1athan mostrato in figura 5

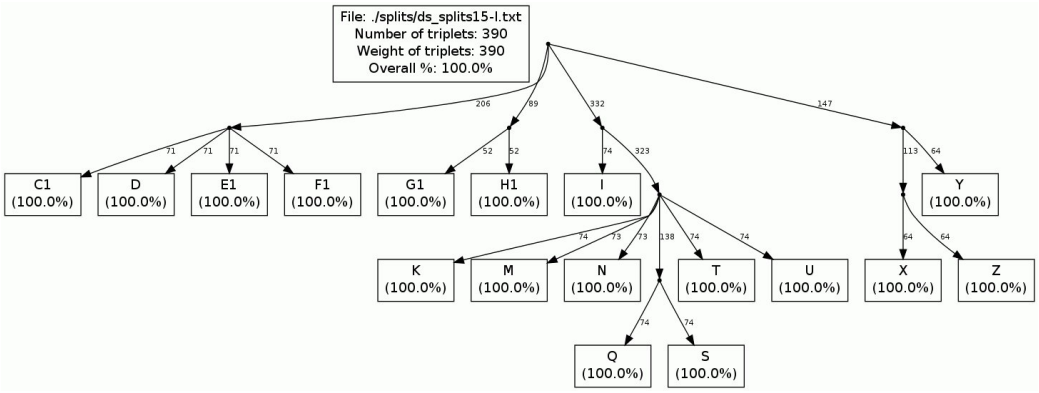


Figura 7: Output di Lev1athan sull'istanza *ds_splits15-l.txt*

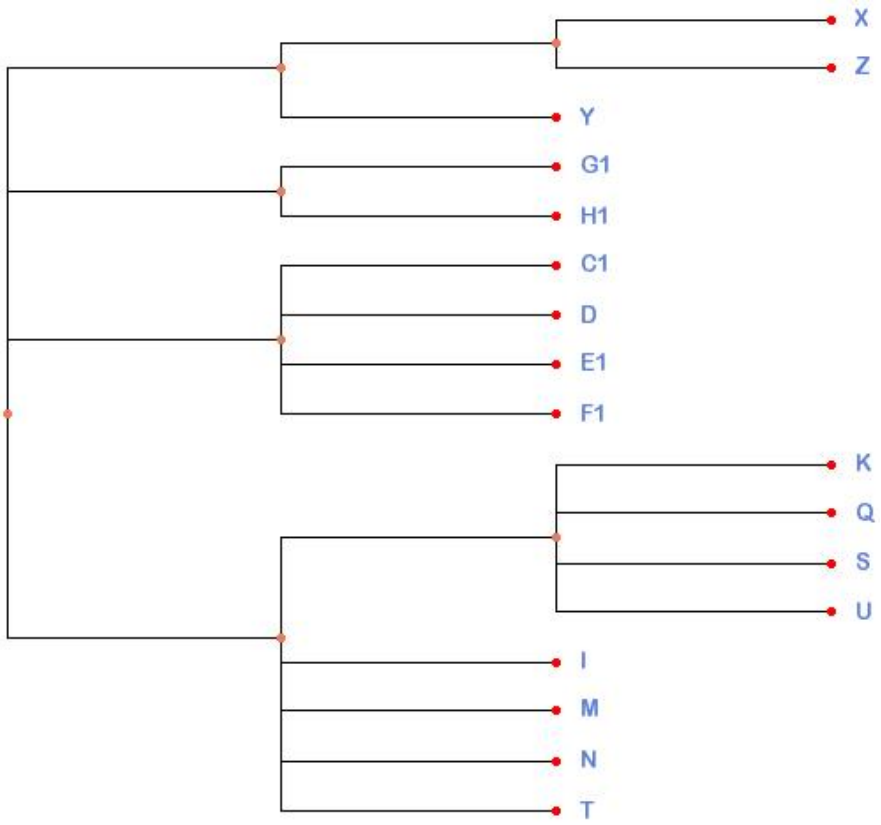


Figura 8: Output del modello MILP sull'istanza *ds_splits15-l.txt*

# 5 Conclusioni e possibili sviluppi

In questo progetto è stato sviluppato un modello di programmazione lineare intera mista in grado di trovare l'albero filogenetico minimale consistente con un insieme di triple radicate. Il problema di ottimizzazione è intrinsecamente difficile (NP-hard), quindi richiede l'uso di euristiche per tagliare lo spazio di ricerca formato da tutti i possibili alberi filogenetici su $n$ specie (che sono in numero esponenziale rispetto a $n$). Il modello presentato ha il vantaggio di garantire l'ottimalità della soluzione trovata e ha dimostrato in alcuni casi di saper trovare soluzioni migliori di quelle fornite dal tool Lev1athan[3]. Lo svantaggio principale del modello riguarda i tempi di esecuzione, che su istanze di una certa dimensione si sono rivelati essere decisamente superiori di quelli di Lev1athan. Il motivo dei lunghi tempi di risoluzione risiede nel fatto che il modello utilizza un numero quadratico di variabili intere, che si rivelano però necessarie per modellare la struttura ad albero.

## 5.1 Estensione del modello alle reti filogenetiche

Le reti filogenetiche sono grafi aciclici radicati in cui possono esistere nodi con due padri e in cui le foglie corrispondono alle specie[10]. Questi grafi sono una struttura più generale degli alberi filogenetici e vengono utilizzati in filogenetica per rappresentare situazioni non rappresentabili mediante alberi; esistono infatti insiemi di triple radicate non compatibili con nessun albero ma compatibili con una rete filogenetica, come ad esempio l'insieme $\{AB|C,\ BC|A\}$ la cui rete è mostrata in figura 9.

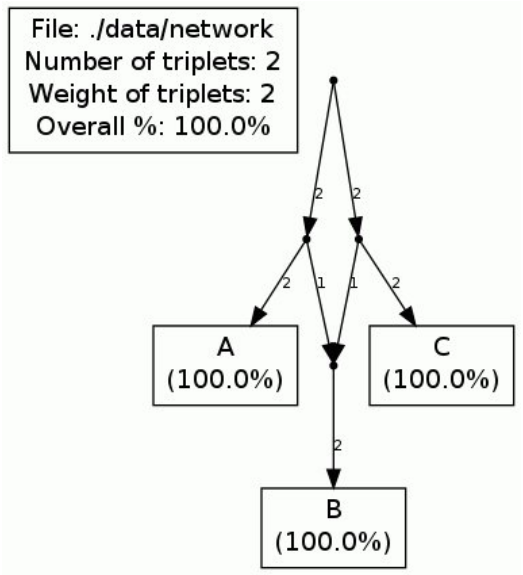


Figura 9: Output di Lev1athan date le triple $AB|C$ e $BC|A$

Il tool Lev1athan può in realtà risolvere problemi più generali di quelli per il quale è stato qui utilizzato in quanto riesce a trovare una *rete* filogenetica minimale consistente con un insieme di triple radicate. Il modello presentato in sezione 2 può essere esteso alle reti filogenetiche ammettendo che il numero di padri di ogni nodo possa variare tra 1 e 2 (anzichè essere vincolato a 1) e modificando il vincolo di consistenza con l'insieme di triple. Questa ultima modifica è la più complessa da apportare e non può essere espressa semplicemente come in sezione 2 a causa del fatto che si possono avere nodi con più di un padre e quindi il calcolo del LCA non verrebbe eseguito in modo corretto. Un possibile (ma costoso) modo per implementarla sarebbe quello diretto (seguendo la definizione): introdurre una matrice a 5 dimensioni $M_{uvijk}$ tale che $M_{uvijk} = 1$ se e solo se esistono i cammini da $u$ a $i$, da $u$ a $j$, da $v$ a $k$ e da $v$ a $u$ (i vincoli da imporre su $M$ sono analoghi

a quelli usati in sezione 2 per $L$) e per ogni tripla radicata $ij|k$ imporre che $\sum_{u \neq v} M_{uvijk} \geq 1$.

# Riferimenti bibliografici


[1] http://en.wikipedia.org/wiki/Newick_format.

[2] http://it.wikipedia.org/wiki/Criterio_della_massima_parsimonia.

[3] LEV1ATHAN Level-1 Heuristic, http://skelk.sdf-eu.org/lev1athan/.

[4] Alfred V. Aho, Yehoshua Sagiv, Thomas G. Szymanski, and Jeffrey D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.

[5] Jakob Fredslund. PHY· FI: fast and easy online creation and manipulation of phylogeny color figures (http://cgi-www.cs.au.dk/cgi-chili/phyfi/go). *BMC bioinformatics*, 7(1):315, 2006.

[6] J. Jansson, R.S. Lemence, and A. Lingas. The complexity of inferring a minimally resolved phylogenetic supertree. *SIAM Journal on Computing*, 41(1):272–291, 2012.

[7] L. Marcus, F. Sven, M. Manja, and P. Sonja. Proteinortho: Detection of (Co-) orthologs in large-scale analysis. *BMC Bioinformatics*, 12.

[8] Daniel Merkle and Nicolas Wieseke. An ILP formulation for fixing estimated orthology matrices. Technical report.

[9] Srinath Sridhar, Fumei Lam, Guy E Blelloch, R Ravi, and Russell Schwartz. Efficiently finding the most parsimonious phylogenetic tree via linear programming. In *Bioinformatics Research and Applications*, pages 37–48. Springer, 2007.

[10] Leo van Iersel, Judith Keijsper, Steven Kelk, Leen Stougie, Ferry Hagen, and Teun Boekhout. Constructing level-2 phylogenetic networks from triplets. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 6(4):667–681, 2009.