Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Magistrale in Informatica

ALGORITMI PER LA DISTRIBUZIONE DEI DATI IN RETI NON CABLATE DI SENSORI

Tesi di Laurea in Algoritmi

Relatore:
Chiar.mo Prof.
ALAN ALBERT BERTOSSI

Presentata da:
LORENZO BELLI

Sessione III
2011/2012

Indice

Lista degli Algoritmi

Sommario

Le reti di sensori non cablate producono una grande quantità di informazioni sotto forma di flusso continuo di dati, provenienti da una certa area fisica. Individualmente, ogni sensore è autonomo, dotato di batteria limitata e possiede una piccola antenna per la comunicazione; collettivamente, i sensori cooperano su un' area più o meno vasta per far giungere i dati generati ad un unità centrale. Poiché la sostituzione delle batterie è spesso un operazione troppo costosa o inattuabile, l'efficienza energetica è considerata una metrica prioritaria durante la progettazione delle reti di sensori non cablate. Si richiede non solo di ridurre le richieste energetiche di ogni singolo nodo, ma anche di massimizzare il tempo di vita dell'intera rete, considerando i costi di comunicazione tra sensori. Ciò ha portato allo studio di come rimuovere le inefficienze energetiche sotto ogni aspetto: dalla piattaforma hardware, al software di base, ai protocolli di comunicazione al software applicativo.

Nella tesi è illustrata una tecnica per il risparmio energetico che consiste nell'applicare memorie fisiche ad alcuni nodi della rete, in modo da accumulare in esse porzioni dei dati generati; successivamente le informazioni possono essere recuperate dall'unità centrale tramite interrogazioni. Questo permette di ridurre il numero di dati trasmessi, e conseguentemente diminuire l'assorbimento energetico della rete. Scopo della presente tesi è individuare algoritmi per determinare la disposizione ottima delle memorie tra i nodi.

Organizzazione della tesi

La tesi è organizzata in cinque capitoli.
Nel Capitolo 1 viene definito formalmente il problema da risolvere, viene specificato il modello di consumo energetico e viene introdotta la terminologia necessaria per lo studio degli algoritmi nei capitoli successivi.
Il Capitolo 2 tratta la risoluzione del problema specificato, supponendo che l'invio dell'interrogazione tra i nodi avvenga con un particolare modello di comunicazione (detto *Modello 1*).
Il Capitolo 3 affronta il medesimo problema applicando un nuovo modello di comunicazione (detto *Modello 2*), che elimina una particolare semplificazione caratteristica del Modello 1. Per il Modello 1 ed il Modello 2 sono descritti algoritmi specifici per particolari topologie di rete: albero generico, albero regolare completo, cammino, caterpillar e caterpillar regolare. Per ogni topologia viene descritto il miglior algoritmo da noi individuato.
Il Capitolo 4 studia il problema con un nuovo modello di comunicazione (*Modello 3*), ne analizza le similitudini con il Modello 2 ed espone alcuni algoritmi specifici per il modello in questione.
Il Capitolo 5 infine riassume i principali risultati ottenuti dallo studio degli algoritmi esposti.

Glossario

Nomenclatura	Descrizione
C_i	Insieme dei nodi figli di i.
$C_{c,h}$	Numero di nodi in un albero regolare completo c-ario con profondità massima h. Definito dall'Equazione 2.1.5.
$Cat(a, b)$	Sottoalbero del grafo caterpillar in input, che comprende tutti i nodi da a a b, tutti i loro figli foglia (compresi i figli foglia di a e di b) ed esclude l'eventuale figlio non foglia di b. Vedi Sezione 2.1.4.
$E(i)$	Costo energetico dell'albero radicato in i per unità di tempo. Definito come $\sum_{i \in T_i} e(i)$.
$E_w(i)$	Costo dell'albero radicato in i, che contiene solo nodi bianchi, i compreso; comprende la risalita dei dati grezzi generati dai nodi di T_i fino al padre di i.
E	Costo energetico totale per unità di tempo, definito come $E = \sum_i e(i)$. Può contenere eventualmente il costo per inviare tutte le informazioni dalla radice verso un suo fittizio nodo padre.
$F(i)$	Numero di figli foglia del nodo i.
$F(i, j)$	Numero di foglie in $Cat(i, j)$. Si noti che $F(0, j) = N'_j - j - 1$. Vedi Sezione 2.1.4.
M'_i	Numero di messaggi grezzi e di risposta scambiati in $Cat(0, i)$; non comprende la risalita fino alla radice delle informazioni generate da $Cat(i + 1, h)$; Vedi Sezione 2.1.4.
M_i	Numero di messaggi grezzi e di risposta scambiati in $Cat(i, h)$, ovvero la risalita fino ad i delle informazioni generate dai nodi in $Cat(i, h)$; Vedi Sezione 2.1.4.
N'_i	Numero di nodi in $Cat(0, i)$; Vedi Sezione 2.1.4.
N_i	Numero di nodi in $Cat(i, h)$; Vedi Sezione 2.1.4.
T_i	Sottoalbero radicato in i; inteso anche come insieme dei suoi nodi.
α	Percentuale (media) di dati richiesti dall'interrogazione; $\alpha \in [0, 1]$.

Nomenclatura	Descrizione
b_i	Costo energetico per l'invio di dati al padre, per unità di dati.
c	Numero massimo di figli di ogni nodo dell'albero.
d_i	Profondità del nodo i, definita come il numero di archi che lo congiungono con la radice.
$e(i)$	Costo energetico che incorre sul nodo i per unità di tempo; definito in 1.1.1.
e_{re}	Costo energetico di ricezione di un'unità di dati.
e_{tr}	Costo energetico di invio di un'unità di dati.
h	Indice dell'ultimo nodo della dorsale in un grafo caterpillar.
i	Si indica con i un nodo dell'albero di input. Occasionalmente useremo il termine 'albero i' per indicare tutto il sottoalbero radicato in i; Vedi anche **Albero i**.
k_u	Numero di nodi neri utilizzati nella colorazione ottima nel caso non limitato.
k	Massimo numero di nodi neri da piazzare nel caso limitato. $k >= 1$.
n	Numero di nodi nell'albero di input. L radice è indicizzata come 0.
r_d	Numero di dati generati per unità di tempo dai sensori.
r_q	Numero (medio) di interrogazioni generate per unità di tempo dalla radice.
s_d	Dimensione del pacchetto dati generato da ogni sensore ogni intervallo r_d di tempo .
s_q	Dimensione (media) del messaggio di interrogazione.
$\|T_i\|$	Numero di nodi contenuti nel sottoalbero T_i.
Albero i	Sottoalbero radicato in i, Vedi anche T_i.
Colorazione	Funzione che associa a tutti i nodi di un albero un colore bianco o nero.
Dati grezzi	Dati trasportati verso la radice che non sono memorizzati nei nodi neri; vengono inviati immediatamente e non in risposta ad un messaggio di interrogazione; sono inviati solamente da nodi bianchi. Detti anche *dati raw*.
Dorsale	Cammino centrale di un caterpillar; contiene $h + 1$ nodi. Denominata anche *backbone*. Vedi Sezione 2.1.4.
Frontiera	Insieme di nodi neri (nel caso non limitato) che hanno figli bianchi nella soluzione ottima.
Messaggi di interrogazione	Mesaggi che trasportano l'interrogazione da propagare a tutti i nodi neri. Hanno dimensione s_q e frequenza r_q.

Nomenclatura	Descrizione
Messaggi di risposta	Messaggi che trasportano dati precedentemente memorizzati in nodi neri. Scaturiscono dai nodi neri solo in risposta ad un messaggio di interrogazione. Hanno dimensione $s_d\alpha$ e frequenza r_q.
Messaggi grezzi	Messaggi che trasportano dati grezzi; possono essere inviati e inoltrati solo da nodi bianchi. Ha dimensione s_d e frequenza r_d.
Nodo bianco	Sensore senza memoria che inoltra i dati verso il padre. Se invia dati da esso appena generati utilizza messaggi grezzi. Inoltra al padre senza modificarli i messaggi provenienti dai figli.
Nodo nero	Sensore com memoria che inoltra i dati verso il padre in risposta ad un messaggio di interrogazione.

Capitolo 1

Formulazione del problema

Per captare informazioni ambientali o monitorare comportamenti è possibile utilizzare una **rete sensoriale**; queste sono composte da sensori, fisicamente distribuiti su un'area, che captano informazioni a intervalli di tempo predefiniti e le inoltrano tra loro per farle giungere ove occorrano. Per una lista di possibili applicazioni è possibile consultare [1], [2] e [3]; per una trattazione molto più ampia si consiglia invece di consultare [4] e [1].

La natura dei dati restituiti dai sensori può essere molteplice, per esempio un unità centrale può chiedere alla rete "Qual'è l'ultima temperatura rilevata?"o "Quali sensori hanno registrato il passaggio di un velivolo?"; noi non faremo particolari assunzioni sulla tipologia di dati acquisiti. Tutte le informazioni generate devono essere memorizzate per una futura consultazione, i dati possono quindi essere memorizzati in un'unità centrale o nei nodi della rete [5] [6]. Se si decide di memorizzare i dati in modo distribuito tra i nodi, possono sorgere diversi problemi. Per primo, ogni sensore ha solo una capacità limitata di memoria che proibisce di mantenere uno storico di tutti i dati generati. Secondo, ogni sensore opera a batteria, quindi i dati vengono persi quando finisce l'energia accumulata, ed inoltre il sensore stesso cessa la sua attività. Terzo, occorre recuperare i dati desiderati in una rete che può essere anche molto vasta. Si deduce quindi che il problema di come distribuire i dati può diventare molto complesso, se teniamo conto che alcune informazioni dovranno percorrere un lungo cammino all'interno della rete per arrivare alla loro destinazione, e di conseguenza molti nodi utilizzeranno energia per inoltrarle, diminuendo così il tempo di vita della rete. Tipicamente, i nodi vicino al pozzo che raccoglie i dati avranno un consumo energetico maggiore, poiché dovranno inoltrare i dati generati da moltissimi sensori [5].

I sensori comunicano tra loro senza fili, quindi ognuno di essi può comunicare con ogni altro sensore all'interno del suo raggio d'azione. Scopo della rete di sensori è far giungere i dati captati ad un unità centrale quando essa li richiede. Viene quindi creato un **albero di comunicazione** tra i sensori, che diventano nodi di tale albero, del quale l'unità centrale (pozzo) è la radice. Ogni sensore avrà dunque un nodo padre e diversi nodi figli. L'invio dei messaggi tra nodi avviene in broadcast, dove un nodo invia un messaggio a tutti i vicini, ed i vicini possono scegliere se ricevere o meno tale messaggio[1]. I messaggi possono essere diretti verso il padre o verso i figli. Nel configurare l'albero di comunicazione vari fattori possono essere presi in considerazione, come ad esempio la stabilità di una connessione, la distanza tra i nodi, la probabilità di perdita di informazione e così via, ma ciò è al di fuori degli interessi di questa tesi. Alcuni esempi su come costruire l'albero di comunicazione si

[1] Con vicini si intendono i nodi adiacenti nell'albero di comunicazione, e non tutti i nodi nel raggio d'azione dell'antenna del trasmettitore.

possono trovare in [7] e [8].

Ogni sensore genera una quantità di dati s_d un numero di volte r_d durante ogni unità di tempo. I sensori normalmente non hanno una memoria, quindi devono inoltrare immediatamente al padre i dati da loro generati e quelli ricevuti dai figli, in modo da farli giungere alla radice. Chiamiamo tali sensori **nodi bianchi**, i dati che generano **dati grezzi** ed i messaggi che li trasportano **messaggi grezzi** (detti anche *messaggi raw*, in inglese). I sensori sono dotati di batteria limitata, nel caso che un nodo esaurisca l'energia di cui dispone si può eventualmente variare l'albero di comunicazione, ma resta essenziale cercare di prolungare per quanto possibile la vita di una rete diminuendone le richieste energetiche. La tecnica qui presentata consiste nel sostituire alcuni sensori con altri **con memoria** detti **nodi neri**. I sensori con memoria non inoltrano i dati immediatamente quando giungono a loro, ma li inviano al padre solo in risposta ad una interrogazione. I nodi neri memorizzano solo gli ultimi dati generati da essi e dai discendenti bianchi, e quindi non mantengono tutto lo storico dei dati generati nel tempo [2]. Per motivi puramente pratici assumeremo sempre che la radice sia un nodo nero. La radice può effettuare **interrogazioni** (dette anche *query*) diverse tra loro alla rete di sensori e tali interrogazioni vengono inoltrate da padre in figlio fino a raggiungere tutti nodi neri, che risponderanno inoltrando verso la radice i dati memorizzati tramite **messaggi di risposta** (anche detti *query reply*). Ai fini dell'analisi, anche se le interrogazioni sono potenzialmente diverse tra loro, supponiamo che mediamente la radice effettui r_q interrogazioni in ogni unità di tempo, e che la dimensione media del messaggio di interrogazione sia s_q unità di dati. Le interrogazioni inoltre richiedono mediamente alla rete una porzione α dei dati, con $\alpha \in (0, 1]$.[3] La Figura 1.0.1 riassume quali tipi di messaggi esistono ed in quali casi essi occorrano.

La definizione dell'albero di comunicazione può avvenire prima o dopo il piazzamento dei nodi neri. Noi ci occuperemo solo del **modello ad albero fisso** in cui l'albero di comunicazione è dato in input e non varia dopo aver stabilito dove piazzare i nodi con memoria. All'opposto nel **modello ad albero dinamico** vengono prima piazzati i nodi neri, e l'albero di comunicazione varia di conseguenza. Tale problema è NP-arduo [7] ed alcuni algoritmi di approssimazione sono già stati proposti [9].

Gli algoritmi da noi studiati investigano i casi in cui l'albero di comunicazione abbia alcune particolari topologie; verrano studiati in particolare i casi di albero generico, albero regolare completo, cammino, caterpillar e caterpillar regolare. È disponibile un algoritmo per il modello ad albero fisso [5], ma utilizza un modello energetico diverso da quello esposto nella presente trattazione (in particolare assume trascurabile il costo di diffusione dell'interrogazione). Altri algoritmi indipendenti dalla topologia sono già stati trattati da *Aly et al.* [10].

Il posizionamento dei nodi neri è estremamente importante in questo modello, infatti una cattiva disposizione potrebbe avere un effetto negativo sul consumo energetico. Il vero vantaggio di usare memorie consiste nel poter richiedere i dati di un sottoalbero solo quando servono, e poter inoltrare solo la porzione α di dati richiesti. Un buon algoritmo di piazzamento deve tenere in considerazione il compromesso tra l'accumulo di dati ed il costo dell'interrogazione. Si noti infatti che se disponiamo un nodo con memoria molto lontano dalla radice, allora l'interrogazione dovrà essere inoltrata a grande distanza.

[2] Un nodo nero non mantiene in memoria i dati generati dai suoi discendenti neri e dai loro discendenti: i messaggi che trasportano tali informazioni sono semplicemente inoltrati al padre.

[3] Se invece sapessimo che la rete deve rispondere ad un solo tipo di interrogazione con $\alpha = 1$ e frequenza r_q, allora potremmo impostare i sensori in modo che generino i dati richiesti ogni $r_d = r_q$ unità di tempo, così da rendere inutile la memorizzazione.

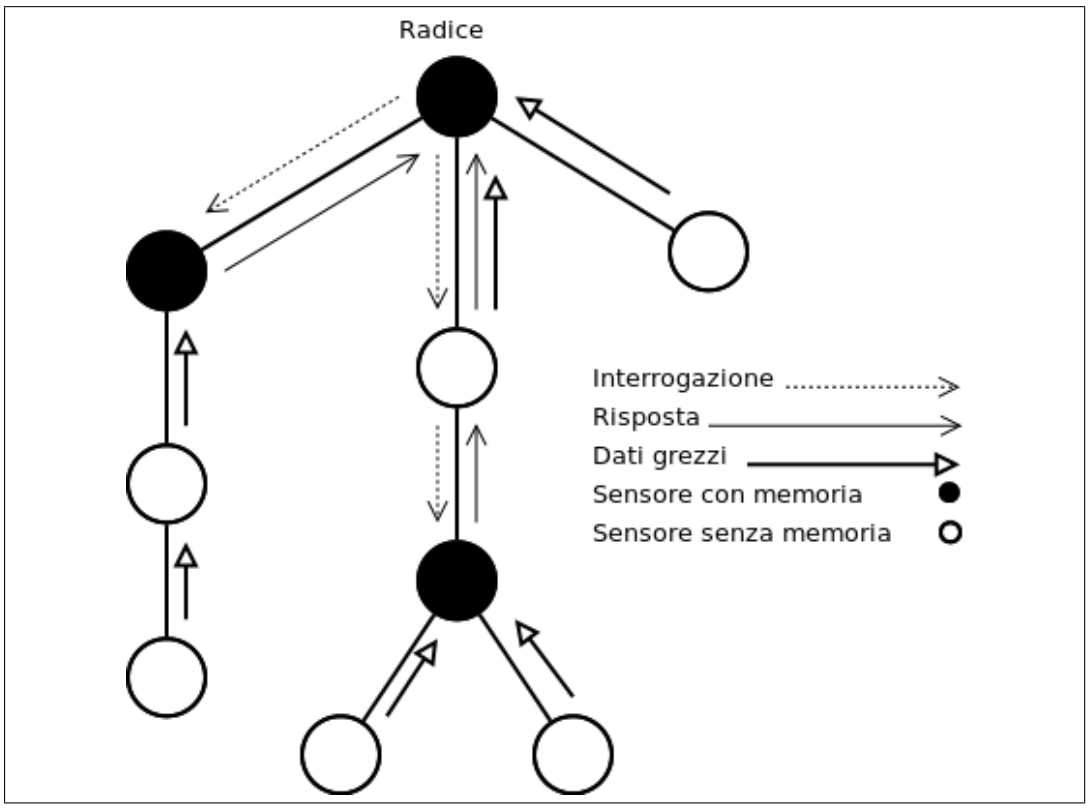


Figura 1.0.1: Esistono 3 tipi di messaggi: i messaggi **di interrogazione** che vengono inoltrati dalla radice verso tutti i nodi neri, i **messaggi di risposta** che vengono inoltrati dai nodi neri verso la radice in risposta a **messaggi di interrogazione** ed i **messaggi grezzi** che trasportano dati non ancora memorizzati in alcun nodo nero.

Tra le possibili metriche che si possono utilizzare per guidare la disposizione dei nodi neri, cercare di massimizzare il tempo di vita della rete sembra essere la migliore (definito come il tempo che passa fin quando un nodo esaurisce la sua batteria); noi useremo un'approssimazione del tempo di vita: il costo energetico totale $\boldsymbol{E}$, ottenuto sommando i costi energetici che incorrono in tutti i nodi in ogni unità di tempo.

Scopo della presente tesi è definire gli algoritmi con cui scegliere la disposizione dei nodi neri all'interno di una rete sensoriale per minimizzare il costo energetico totale in un modello ad albero fisso.

Sono trattati due possibili casi: nel caso **non limitato** si suppone di avere a disposizione un numero qualsiasi di nodi neri, mentre nel **caso limitato** abbiamo a diposizione un numero limitato $\boldsymbol{k}$ di nodi neri da piazzare (se ne possono piazzare di meno se occorre). Si osservi che il nodo radice viene sempre considerato nero, e conteggiato nel valore k in input (in altre parole possiamo ridurci a dover scegliere dove piazzare solo $k-1$ nodi neri).

1.1 Modello di costo

Presentiamo ora un modello per analizzare il costo energetico di una rete, originariamente mostrato in [7]; useremo questo modello negli algoritmi presentati successivamente, tranne dove diversamente indicato. Assumiamo che il costo energetico sia proporzionale solo alla dimensione del pacchetto dati scambiato, e non dipenda dalla distanza fisica tra i nodi (o, in altre parole, che la distanza tra i nodi sia costante), e quindi identico per tutti gli archi. Nel caso in cui un pacchetto venga perso durante una trasmissione il mittente può rispedirlo; assumiamo comunque che la probabilità di ritrasmissione sia identica per tutti gli archi, e di conseguenza il costo energetico di trasmissione è ancora proporzionale alla dimensione del pacchetto dati.

I costi energetici per trasmettere un'unità di informazione, e per riceverla, sono rispettivamente $\boldsymbol{e_{tr}}$ e $\boldsymbol{e_{re}}$, con i vincoli $e_{tr} \geq 0, e_{re} \geq 0, e_{tr} + e_{re} > 0$. Il costo di ricezione dei messaggi viene assegnato al mittente senza ripercussioni sul costo totale, dunque la trasmissione di un unità di dati dal nodo i al nodo j non fa consumare energia al nodo j, mentre invece il costo energetico per il nodo i è

$$\begin{cases} e_{tr} + e_{re} & \text{se } j \text{ è padre di } i \\ e_{tr} + e_{re} v_i & \text{altrimenti} \end{cases}$$

dove v_i è il numero di nodi che ricevono il messaggio di i.
Per una migliore presentazione normalizziamo il costo energetico per un fattore $e_{tr} + e_{re}$, così facendo il costo energetico per inviare un'unità di informazione da i a j diventa:

$$\begin{cases} 1 & \text{se } j \text{ è padre di } i \\ b_i = \frac{e_{tr} + e_{re} v_i}{e_{tr} + e_{re}} & \text{altrimenti} \end{cases} \tag{1.1.1}$$

In questo modo il costo per la risalita dei dati verso la radice diventa =1, semplificando così le formule che tratteremo; chiamiamo inoltre $\boldsymbol{b_i}$ il costo energetico per l'invio da i verso i suoi figli di un unità di informazione. Esplicitiamo che solo i messaggi di interrogazione viaggiano dalla radice verso le foglie, dunque questi sono gli unici messaggi che avranno un costo energetico b_i potenzialmente diverso da 1.

Sia $\boldsymbol{i}$ un nodo dell'albero di comunicazione, e $\boldsymbol{T_i}$ il suo sottoalbero radicato in i, con $\boldsymbol{|T_i|}$ si indichino il numero di nodi in T_i. Definiamo $\boldsymbol{e(i)}$ come il costo energetico di un nodo per unità di tempo, che include il trasferimento dei dati in risposta all'interrogazione (se i è nero), l'inoltro dell'interrogazione se i ha discendenti neri, l'invio verso il padre dei dati non memorizzati (se i è bianco), e l'inoltro al padre dei dati generati dai discendenti (se i è bianco non foglia). Possiamo quindi definire matematicamente $e(i)$ per casi:

Definizione 1.1.1.

Caso 1. $e(i) = |T_i| r_d s_d$, se i è bianco e non ha discendenti neri; allora non occorre inoltrare l'interrogazione, e tutti i dati generati dai $|T_i|$ nodi in T_i devono essere inviati al padre.

Caso 2. $e(i) = |T_i| r_q \alpha s_d$, se i è nero e non ha discendenti neri; l'interrogazione non deve essere inoltrata, e tutti i dati generati dai sensori in T_i sono restituiti come messaggi di risposta.

Caso 3. $e(i) = |T_i| r_q \alpha s_d + b_i r_q s_q$, se i è nero ed ha dei discendenti neri; i dati generati dai sensori in T_i sono restituiti in risposta all'interrogazione, e l'interrogazione deve essere inoltrata.

Caso 4. $e(i) = (d_1+1)r_d s_d + b_i r_q s_q + r_q \alpha d_2 s_d$, se i è un nodo bianco ed ha dei discendenti neri.
Tra i $|T_i|-1$ discendenti di i esattamente d_i sono bianchi e non hanno nodi neri nel loro cammino verso i; d_2 è invece il numero di nodi neri in T_i sommato al numero di nodi bianchi che hanno antenati neri in T_i; si noti che $d_1 + d_2 = |T_i| - 1$.

Definiamo poi $\boldsymbol{E(i)}$ come il costo di un sottoalbero T_i, dove

$$E(i) = \sum_{i \in T_i} e(i) \tag{1.1.2}$$

Possiamo ora definire il costo energetico totale $E = E(0)$ come

$$E = \sum_{i} e(i) \tag{1.1.3}$$

Così espresso E comprende anche il costo di uscita delle informazioni dal nodo radice verso un ipotetico padre; vedremo che tale costo può essere anche omesso poiché costante.

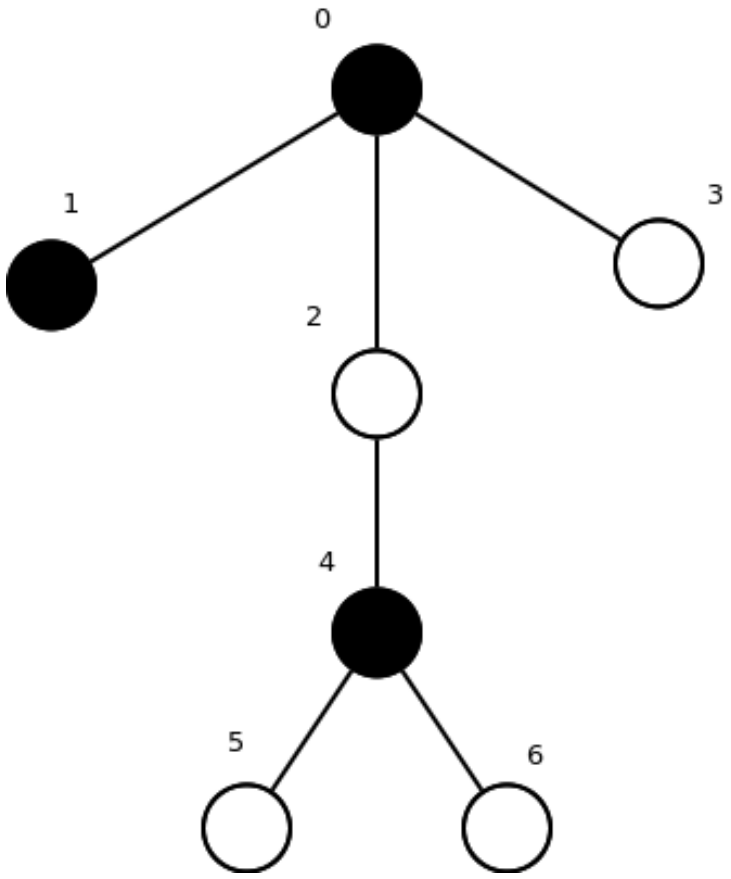
Da ciò che abbiamo visto fin'ora il costo di invio di un messaggio grezzo è di $r_d s_d$, mentre il costo di invio di un messaggio di risposta è di $r_q s_d \alpha$. Il principale vantaggio di utilizzare nodi con memoria risiede nel fatto che i messaggi di risposta avranno tipicamente un costo inferiore ai messaggi grezzi, per cui esplicitiamo la seguente affermazione banale

Teorema 1.1.2. *Se $r_q s_d \alpha > r_d s_d$ allora il costo di invio di un messaggio di risposta è mggiore al costo di un messaggio grezzo, quindi non è in alcun modo vantaggioso memorizzare i dati tra i nodi. Viceversa quando $r_q s_d \alpha > r_d s_d$ il costo di un messaggio di risposta che trasporta l'informazione generata da un solo nodo αs_d è inferiore a quello di un messaggio grezzo r_d.*

È bene osservare, come conseguenza banale del Teorema 1.1.2:

Teorema 1.1.3. *Se un nodo i è nero, allora possiamo rendere nero uno qualsiasi tra i nodi bianchi che ricevono l'interrogazione (come gli antenati di i o i fratelli dei suoi antenati), senza aumentare il costo dell'albero, ma anzi diminuendolo.*

Mostriamo in Figura 1.1.1 un esempio su come calcolare il costo di un albero. In questo esempio l'interrogazione viene inviata solo ai nodi neri o ai nodi bianchi con discendenti neri; come vedremo tale caratteristica varierà con la presentazione dei modelli di comunicazione nei capitoli successivi.



(a) Grafo di esempio

nodo i	invia interrogazione?	riceve interrogazione?	$e(i)$	m_i
0	si		$7r_q s_q \alpha + \frac{e_{tr}+2e_{re}}{e_{tr}+e_{re}}$	7
1		si	$r_q s_d \alpha$	1
2	si	si	$r_d s_d + 3r_q s_q \alpha + \frac{e_{tr}+e_{re}}{e_{tr}+e_{re}}$	4
3		(*)	$r_d s_d$	1
4		si	$3r_q s_d \alpha$	3
5			$r_d s_d$	1
6			$r_d s_d$	1

(b) Valori per ogni nodo.

Figura 1.1.1: Esempio su come calcolare il costo di un albero. Si può ottenere il costo toale con $E = \sum_{i=0}^{6} e(i)$.
(*) Il nodo 3 riceve l'interrogazione se consideriamo il primo modello trattato nel Capitolo 2 e non la riceve se consideriamo il modello 2 trattato nella Sezione 3. m_i indica il numero di messaggi grezzi e di risposta inviati da quel nodo verso il padre.

Capitolo 2

Modello 1

Il primo modello di comunicazione trattato ha la seguente caratteristica: *l'invio dell'interrogazione da parte nel nodo i avviene in broadcast, e tutti i figli di i la ricevono.* Il costo di trasmissione dell'interrogazione derivante dall'Equazione 1.1.1 è dunque

$$b_i = \frac{e_{tr} + e_{re}|C_i|}{e_{tr} + e_{re}}$$

Definiamo il costo di un sottoalbero come segue:

Definizione 2.0.4. Sia $\boldsymbol{E(i)}$ il costo del sottoalbero T_i definito come $E(i) = \sum_{i \in T_i} e(i)$. Tale costo comprende la diffusione dell'interrogazione da i ai discendenti, e la risalita di informazioni da tutti i nodi di T_i fino al padre di i.

Questo permette di enunciare il seguente teorema:

Teorema 2.0.5. *Sia $E(i)$ il costo di un albero radicato in i, se $\alpha r_q \geq r_d$ allora i deve essere un nodo bianco per minimizzare $E(i)$, altrimenti i deve essere un nodo nero.*

Intuitivamente, se $\alpha r_q \geq r_d$, o, riscrivendo, $\alpha r_q s_d \geq r_d s_d$ significa che le interrogazioni richiedono più dati di quanti ne vengono prodotti in un certo lasso di tempo, e quindi alcune interrogazioni in successione otterrebbero gli stessi risultati dai sensori; preferiamo quindi non utilizzare nodi con memoria.
La dimostrazione completa si trova in [7] ed è riportata di seguito.

Dimostrazione. Compariamo il costo di due alberi indentici in ogni aspetto, differenziati solo per la radice: il primo avrà radice bianca, il secondo avrà radice nera. Siano rispettivamente E_1 e E_2 i loro costi. Siano altresì $e(1)$ ed $e(2)$ il costo dei loro rispettivi nodi radice come definiti 1.1.1. Avremo dunque $E_1 - E_2 = e(1) - e(2)$. Per dimostrare il teorema è sufficiente dimostrare:

$$e(1) - e(2) = \begin{cases} \leq 0 & \text{se } \alpha r_q \geq r_d \\ > 0 & \text{altrimenti} \end{cases}$$

Consideriamo due casi:

Caso 1. entrambe le radici non hanno discendenti neri; allora per definizione di $e(i)$ otteniamo

$$e(1) - e(2) = |T_i| r_d s_d - r_q \alpha |T_i| s_d = |T_i| s_d (r_d - \alpha r_q) = \begin{cases} \leq 0 & \text{se } \alpha r_q \geq r_d \\ > 0 & \text{altrimenti} \end{cases}$$

Caso 2. entrambe le radici hanno almeno un discendente nero; allora secondo la definizione 1.1.1 di $e(i)$ abbiamo:

$$e(1) - e(2) = ((d_1 + 1)r_d s_d + b_i r_q s_q + r_q \alpha d_2 s_d) - (r_q \alpha |T_i| s_d + b_i r_q s_q) =$$
$$= (d_1 + 1)s_d(r_d - \alpha r_q) = \begin{cases} \leq 0 & \text{se } \alpha r_q \geq r_d \\ > 0 & \text{altrimenti} \end{cases}$$

dove d_1 è il numero di nodi bianchi (nel primo albero) discendenti della radice che non hanno nodi neri nel cammino che li congiunge alla radice, mentre d_2 è il numero di nodi neri (nel primo albero) sommato al numero di nodi bianchi che hanno un antenato nero.

□

Conseguenza diretta del Teorema 2.0.5 è:

Corollario 2.0.6. *Se $\alpha r_q \geq r_d$ allora ogni nodo (ad eccezione della radice che è sempre nera) dell'albero sarà bianco per ottimizzarne il costo.*

Dimostrazione. Per assurdo supponiamo che valga $\alpha r_q \geq r_d$ e che nella colorazione Z che ottimizza il costo E dell'albero il nodo i sia nero; allora per il Teorema 2.0.5 il costo ottimo del sottoalbero T_i lo si ottiene ponendo i bianco. Ponendo dunque i bianco all'interno di Z otterremmo un costo $E' < E$, [1] e ciò è assurdo poiché E era il costo ottimo. □

Si noti che nel caso $\alpha r_q = r_d$ poniamo come soluzione ottima quella con solo nodi bianchi, poiché come vedremo nel caso limitato, le memorie hanno un costo hardware, ed è quindi preferibile utilizzarne meno a parità di energia.

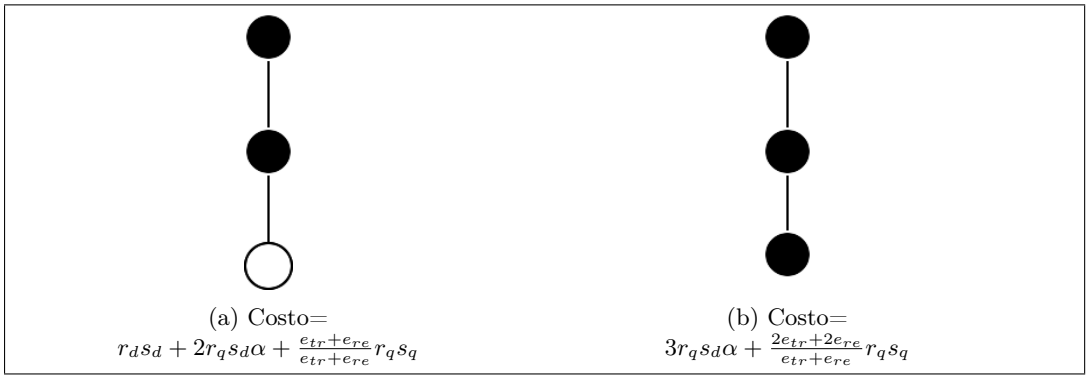
Se dunque ci viene posto il problema di trovare la disposizione ottima per un certo numero di nodi neri in un dato albero, dovremmo prima verificare la condizione $\alpha r_q \geq r_d$: se è soddisfatta sapremo che solo la radice dovrà essere nera. Si noti che il contrario non vale, cioè non è vero che se il costo ottimo di un albero si ottiene ponendo la sola radice nera, allora vale $\alpha r_q \geq r_d$: ad esempio se la dimensione dell'interrogazione fosse enorme $s_q = +\infty$ è banale osservare che non dovremmo avere nodi neri per ottimizzare il costo, ma può comunque non valere la relazione $\alpha r_q \geq r_d$.

Data questa premessa tutti gli algoritmi ed i teoremi che presenteremo in seguito investigheranno il solo caso $\alpha r_q < r_d$ che sarà assunto vero.

[1] Eventualmente tale sostituzione farà in modo che l'interrogazione non sia più necessaria diminuendo ulteriormente il costo e non inficiando la dimostrazione

2.1 Non limitato

Ci poniamo dapprima nel caso apparentemente più semplice, supponiamo quindi di avere a disposizione un numero non limitato di nodi neri; chiameremo tale problema **caso non limitato**. Potremmo essere tentati di far diventare neri tutti i nodi di un albero per minimizzarne il costo, ma in questo modo l'interrogazione dovrà raggiungere una maggiore profondità, facendo aumentare di conseguenza il costo totale. Un controesempio banale è mostrato nella Figura 2.1.1.



(a) Costo= $r_d s_d + 2r_q s_d \alpha + \frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} r_q s_q$

(b) Costo= $3r_q s_d \alpha + \frac{2e_{tr}+2e_{re}}{e_{tr}+e_{re}} r_q s_q$

Figura 2.1.1: Controesempio che dimostra come utilizzare solo nodi neri non sia sempre la scelta migliore. È infatti possibile trovare dei valori per le variabili in modo che il costo di (a) sia inferiore a quello di (b), per esempio l'assegnamento $r_d = s_d = r_q = 1, s_q = 8, \alpha = 1/2, e_{re} = 0, e_{tr} = 1$ rispetta tale proprietà.

Enunciamo ora i teoremi applicabili al caso non limitato:

Teorema 2.1.1. ***Teorema degli antenati***
Nella colorazione ottima di un albero, se i è un nodo nero, allora esso avrà tutti gli antenati neri.

Questo è banalmente vero: se i avesse un antenato bianco j, esso deve comunque ricevere l'interrogazione ed inoltrarla per farla giungere ad i. Dunque facendo diventare j nero non avremmo nessun costo aggiuntivo per la diffusione dell'interrogazione. Avremmo invece una diminuzione delle informazioni inoltrate da j, che si riducono a $|T_i| r_q s_d \alpha$; tale costo è il minimo ottenibile perché supponiamo di essere nel caso $\alpha r_q < r_d$. È anche conseguenza diretta del Teorema 1.1.2.

Vale inoltre:

Teorema 2.1.2. ***Teorema dei fratelli***
Se nella colorazione ottima di un albero il nodo i è nero, allora anche tutti i suoi fratelli sono neri.

Dimostrazione. Ricordiamo che nel primo modello, l'interrogazione viene inviata ad ogni figlio, quindi nel nostro caso viene inviata dal padre di i a tutti i fratelli di i; questi possono essere quindi trasformati in nodi neri senza costi aggiuntivi poiché ricevono già l'interrogazione. Essi inoltre invieranno dati solo in risposta ad una interrogazione, e poiché $\alpha r_q < r_d$ questo comporta una sicura riduzione di costo. □

Dai precedenti teoremi deduciamo che:

Corollario 2.1.3. ***Secondo teorema dei fratelli***
Nella colorazione ottima, ogni nodo ha tutti i figli neri oppure ha tutti i figli bianchi.

Si evince inoltre il corollario:

Corollario 2.1.4. ***Teorema della frontiera***
La coloraziona ottima di un albero, avrà un insieme di nodi neri adiacenti alla radice, seguiti (eventualmente) da nodi bianchi a maggior profondità. Chiamiamo inoltre l'insieme dei nodi neri con padre nero e solo discendenti bianchi ***frontiera****.*

Tutte le soluzioni appariranno dunque come in Figura 2.1.2.

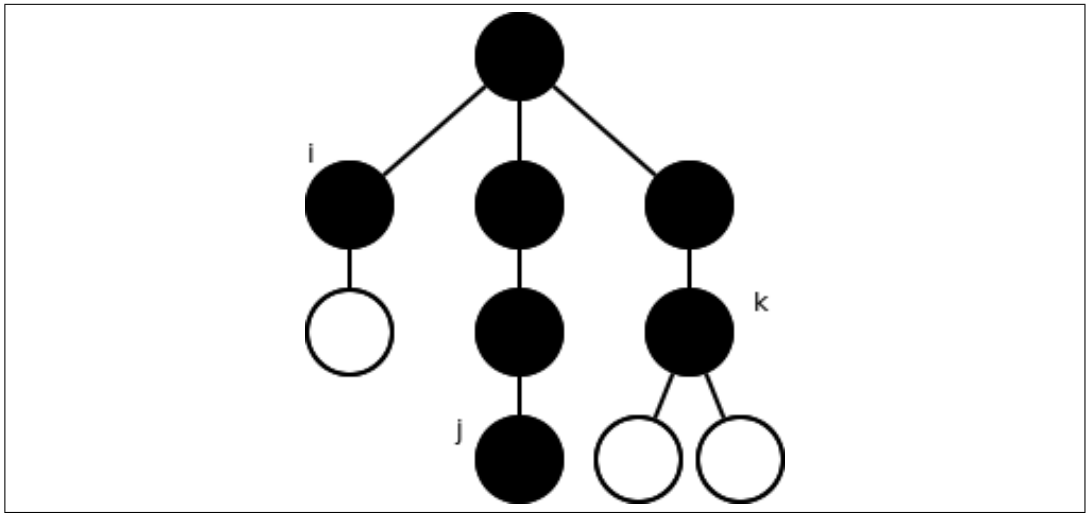


Figura 2.1.2: Esempio di soluzione nel caso non limitato. La frontiera che separa i nodi bianchi dai neri è composta dai nodi $\{i, j, k\}$

2.1.1 Albero

Presentiamo per primo l'algoritmo per risolvere il problema non limitato su un albero generico. L'algoritmo è stato proposto da *Bo Sheng et al.* [7] e servirà come punto di partenza per sviluppare poi gli altri algoritmi.

Ricordiamo il Teorema 2.1.3, secondo il quale i figli di un nodo sono tutti bianchi (e dunque hanno solo discendenti bianchi per il Teorema 2.1.1) oppure tutti neri. Nel caso un nodo abbia solo figli bianchi l'interrogazione non deve essere inoltrata, quindi il suo costo deve essere ignorato. Presentiamo ora un algoritmo che utilizza la tecnica di programmazione dinamica. Sulla base di ciò definiamo $E^*(i)$ come la minima energia (l'ottimo) richiesta dall'albero T_i dove i è nero, che comprende la discesa dell'interrogazione da i alle foglie, e la risalita di informazioni dalle foglie fino al padre di i. Se i è una foglia, allora il costo minimo si ottiene ponendo i nero, e quindi $E^*(i) = r_q s_d \alpha$. Se invece $|C_i| > 0$, il minimo si ottiene considerando due casi: nel primo caso tutti i discendenti sono bianchi, mentre nel secondo avremo tutti i figli neri (e quindi invieremo l'interrogazione), e la definizione si ottiene per ricorrenza:

Definizione 2.1.5.
se i è una foglia allora $E^*(i) = r_q s_d \alpha$, altrimenti

$$E^*(i) = \min \left\{ r_q \alpha |T_i| s_d + \sum_{j \in C_i} E_w(j), \quad r_q \alpha |T_i| s_d + b_i r_q s_q + \sum_{j \in C_i} E^*(j) \right\}$$

dove $\boldsymbol{E_w(i)}$ è il costo di T_i con tutti i nodi bianchi, i compreso.

L'algoritmo non risolve il caso $\alpha r_q \geq r_d$ poiché banale per il Teorema 2.0.6; il caso $\alpha r_q < r_d$ è affrontato tramite programmazione dinamica usando la Definizione 2.1.5. L'algoritmo procede in post-ordine, dalle foglie fino alla radice: esso assume inizialmente che tutti i nodi siano neri, e trasforma in bianchi tutti i nodi appartenenti a certi sottoalberi, fino a raggiungere la colorazione ottima che avrà una certa frontiera. I nodi sono indicizzati da $n-1$ a 0 in postordine, e la radice avrà indice 0. Una tabella $E^*[n-1, \ldots, 0]$ viene utilizzata per memorizzare i costi di tutti i sottoableri radicati nei nodi $i = n-1, \ldots, 0$. Al termine della computazione $E^*[0]$ conterrà il costo minimo di tutto l'albero di input. Manteniano inoltre una seconda tabella $E_w[n-1, \ldots, 0]$ che contiene il costo di tutti i sottoalberi con solo nodi bianchi.

2.1.1.1 Preudocodice

Forniamo ora lo pseudocodice dell'algoritmo.

Algoritmo 1: Modello 1, non limitato, albero generico

```
1  Poni la radice nera;
2  foreach nodo foglia i do
3  |  Poni i nero;
4  |  E*[i] = r_q α s_d;
5  |  E_w[i] = r_d s_d;
   end
6  foreach nodo i non foglia, non radice, in postordine do
7  |  Poni i nero;
8  |  cost_1 = r_q α s_d |T_i| + b_i r_q s_q + Σ_{j∈C_i} E*[j];
9  |  cost_2 = r_q α s_d |T_i| + Σ_{j∈C_i} E_w[j];
10 |  E*[i] = min {cost_1, cost_2};
11 |  E_w[i] = |T_i| r_d s_d + Σ_{j∈C_i} E_w[j];
12 |  if cost_1 ≥ cost_2 then
13 |  |  Poni bianchi tutti i discendenti neri di i;
   |  end
   end
```

I due cicli sono eseguiti ciascuno $O(n)$ volte e ad ogni iterazione viene utilizzato tempo $O(c)$ per calcolare una sommatoria. Mostriamo ora come sia possibile rendere costante il tempo di calcolo delle sommatorie nelle righe 8, 9 e 11.
Siano $L(i) = \sum_{j \in C_i} E^*(j)$ e $L_w(i) = \sum_{j \in C_i} E_w(j)$. Ad ogni iterazione viene calcolato un valore $E^*(i)$ che sommiamo al valore $L(i')$ del padre (i' padre di i); similmente aggiungiamo $E_w(i)$ al valore $L_w(i')$. Per fare ciò utilizziamo due tabelle supplementari $L[0, \ldots, n-1]$ e $L_w[0, \ldots, n-1]$. Riscriviamo dunque l'algoritmo 1 con questa ottimizzazione.

Algoritmo 2: Modello 1, non limitato, albero generico, ottimizzato

```
1  Poni la radice nera;
   // Inizializzazione
2  for i = 0 to n − 1 do
3  |  L[i] = 0;
4  |  L_w[i] = 0;
5  end
6  foreach leaf i do
7  |  Poni i nero;
8  |  E*[i] = r_q α s_d;
9  |  E_w[i] = r_d s_d;
10 |  L[i'] = L[i'] + E*[i];
11 |  L_w[i'] = L_w[i'] + E_w[i];
12 end
13 foreach nodo i non foglia, non radice, in postordine do
14 |  Poni i nero;
15 |  cost_1 = r_q α s_d |T_i| + b_i r_q s_q + L[i];
16 |  cost_2 = r_q α s_d |T_i| + L_w[i];
17 |  E*[i] = min{cost_1, cost_2};
18 |  E_w[i] = |T_i| r_d s_d + L_w[i];
19 |  if cost_1 ≥ cost_2 then
20 |  |  Poni bianchi tutti i discendenti neri di i;
21 |  end
22 |  L[i'] = L[i'] + E*[i];
23 |  L_w[i'] = L_w[i'] + E_w[i];
24 end
```

I cicli più esterni sono eseguiti ciascuno $O(n)$ volte, ed ogni iterazione utilizza tempo costante. Inoltre ogni nodo nero viene trasformato in bianco al più una sola volta. Dunque l'algoritmo ha costo $O(n)$.

2.1.1.2 Dimostrazione di correttezza ed ottimalità

Dimostrazione. Dimostriamo che $E^*(i)$ sia ottimo e ben definito.
La dimostrazione procede per induzione strutturale sul nodo i.
Se i è foglia allora ponendolo nero ne ottimizziamo il costo, dunque $E^*(i) = r_q s_d \alpha$; infatti se lo ponessimo bianco avremmo $r_d s_d$ ed ovviamente $r_d s_d > r_q s_d \alpha$ per il Corollario 2.0.6.
Se i non è foglia, procediamo per casi sulla colorazione ottima Z dell'albero T_i (che ha sempre i nero). L'ipotesi induttiva è che $E^*(j)$ sia ottimo e ben definito per tutti i T_j discendenti di i.

Caso 1. Se i in Z ha solo discendenti bianchi, allora il costo ottimo si ottiene sommando i costi dei sottoalberi $T_j, j \in C_i$ completamente bianchi, ed aggiungendo il termine per l'invio delle informazioni da i al padre. Tale formula è esattamente quella riportata nel primo caso della Definizione 2.1.5. In questo caso l'algoritmo non sceglierà mai il secondo caso, poiché essendo il costo di una colorazione diversa da quella ottima avrà costo maggiore del primo.

Caso 2. Se i in Z ha tutti i figli neri, allora permettiamo anche che i sottoalberi T_j abbiamo un qualsiasi numero di neri. Ogni albero $T_j, j \in C_i$ ha una colorazione ottima (cioè che ottimizza $E^*(j)$) identica a quella che ha in Z. Se così non fosse, un

albero T_j avrebbe colorazione ottima diversa da quella in Z, e sarebbe allora possibile sostituire tale colorazione in Z ottenendone una nuova Z' con costo minore di Z, ma ciò è impossibile poiché Z era ottima. Dunque i sottoalberi T_j hanno costo $E^*(j)$ in Z. Per lo stesso motivo inoltre $E^*(i)$ è ottimo, poiché se non fossero ottimi tutti i valori $E^*(j)$, allora non sarebbe ottimo $E^*(i)$. Il costo $E^*(i)$ si ottiene quindi sommando i costi dei suoi sottoalberi, il costo per la diffusione dell'interrogazione a tutti i figli, ed il costo di invio del messaggio di risposta al padre di i. Tale formula è esattamente quella esposta nella Definizione 2.1.5. Siamo inoltre sicuri che l'algoritmo non scelga mai il primo caso, poiché esso rappresenta il costo di T_i con una colorazione non ottima.

□

2.1.2 Albero regolare completo

Cerchiamo ora un algoritmo per i casi in cui l'albero di input abbia una particolare topologia. Supponiamo di avere un albero regolare e completo, dove ogni nodo ha esattamente c figli con $c \geq 2$, e tutte le foglie si trovano alla medesima profondità.
Due sottoalberi radicati alla stessa profondità hanno identica topologia, dunque avranno la stessa colorazione ottima. Considerato ciò, ricordiamo anche che tutti i fratelli avranno lo stesso colore nell'ottimo (per il Teorema 2.1.2 dei fratelli): ne segue che tutti i nodi che formano la frontiera avranno la stessa profondità. Possiamo dunque cercare iterativamente la profondità della frontiera e calcolare il costo di conseguenza. La Figura 2.1.3 mostra un esempio di albero regolare con $c = 3, n = 7$.

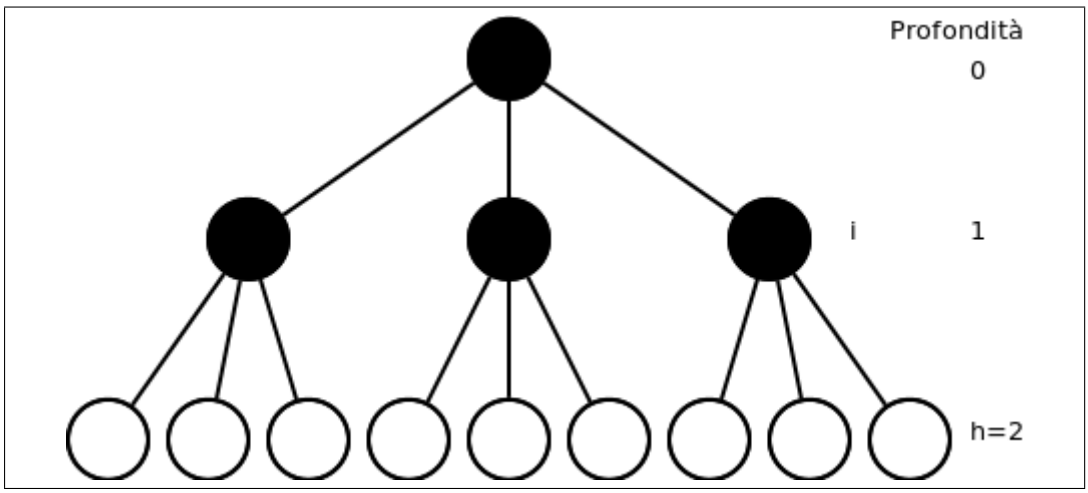


Figura 2.1.3: Esempio di albero regolare con $n = 13$, $c = 3$.

Definiamo il costo totale dell'albero in questo modo:

Definizione 2.1.6. sia $E(i)$ il costo dell'albero quando i nodi della frontiera sono tutti alla profondità i. Non comprende il costo di invio dei dati dalla radice verso il suo ipotetico padre.

Il costo ottimo E dell'albero si trova utilizzando il minimo tra tutti i possibili $E(i)$:

$$E = \min_{i \in [0,h]} \{E(i)\} \tag{2.1.1}$$

Definiamo inoltre le seguenti variabili:

$h = \lfloor \log_c n \rfloor$ profondità dell'albero in ingresso.

$T_i =$ un albero rappresentante tutti i sottoalberi con radice a profondità i.

$M_i =$ numero di messaggi scambiati all'interno dell'albero T_i, generati dai suoi nodi e diretti verso la radice i; non comprende i messaggi di interrogazione.

$R_i =$ numero di messaggi che trasportano i dati generati dai nodi con profondità al più i fino alla radice 0 dell'albero.

$C_{c,h} =$ numero di nodi in un albero regolare completo c-ario con profondità massima h.

Ricordiamo inoltre che in un albero regolare completo il numero di nodi alla profondità i è c^i, mentre il numero di nodi con profondità al più i è $c^{i+1} - 1$.
$E(i)$ si ottiene sommando:

- $c^i M_i r_d s_d$: la risalita dei messaggi grezzi generati dai nodi dei c^i alberi T_i fino alla profondità i;
- $c^i(|T_i| - 1)r_q s_d \alpha$: la risalita, dalla profondità i alla radice, delle informazioni generate dai $|T_i| - 1$ nodi bianchi dei c^i alberi T_i;
- $R_i r_q s_d \alpha$: il costo dei R_i messaggi di risposta che trasportano dati generati da nodi con profondità al più i;
- $\frac{C_{c,i-1}e_{tr}+(C_{c,i}-1)e_{re}}{e_{tr}+e_{re}} r_q s_q$: la trasmissione dell'interrogazione da parte di $C_{c,i-1}$ nodi (vengono esclusi i nodi al livello i), e la ricezione della stessa da parte di $C_{c,i} - 1$ nodi (viene esclusa la radice).

Riassumendo otteniamo

$$E(i) = c^i M_i r_d s_d + c^i(|T_i| - 1)r_q s_d \alpha + R_i r_q s_d \alpha + \frac{C_{c,i-1}e_{tr} + (C_{c,i} - 1)e_{re}}{e_{tr} + e_{re}} r_q s_q \quad (2.1.2)$$

Cerchiamo ora di calcolare i termini non espliciti. Il numero di nodi all'interno di un albero T_i si può calcolare con:

$$|T_i| = c^{h-i+1} - 1$$

M_i rappresenta il numero di messaggi scambiati all'interno di T_i; osserviamo che un dato generato da un nodo a profondità j dovrà attraversare esattamente j archi per giungere alla radice, dunque si può calcolare M_i sommando il numero di nodi ad ogni livello, moltiplicati per la loro profondita:

$$M_i = \sum_{j=1}^{h-i} j c^j = \frac{c^{1-i}\left(hc^{h+1} - ic^{h+1} - c^i - hc^h - (1-i)c^h\right)}{(c-1)^2} \quad (2.1.3)$$

la giustificazione matematica di questa ultima eguaglianza si può trovare in [11, Eq.(3)].

Il termine R_i indica il numero di messaggi scambiati tra i nodi a profondità minore o uguale ad i e portanti informazioni generate dai soli nodi con profondità al più i. Questo può essere calcolato in modo simile a M_i:

$$R_i = \sum_{j=1}^{i} j c^j = \frac{ic^{2+i} - (i+1)c^{i+1} + c}{(c-1)^2} \quad (2.1.4)$$

Notiamo inoltre che R_i si può ottenere anche sottraendo dal numero totale di messaggi scambiati M_0 il numero di messaggi scambiati in ogni sottoalbero $c^i M_i$ e sottraendo la risalita delle informazioni dai nodi al livello i fino alla radice, $c^i|T_i|i$, per cui:

$$R_i = M_0 - c^i M_i - c^i |T_i| i$$

Possiamo calcolare $C_{c,h}$ come somma dei nodi ad ogni profondità:

$$C_{c,h} = c^0 + c^1 + \ldots + c^h = \sum_{i=0}^{h} c^i = \sum_{i=0}^{\infty} c^i - \sum_{i=h+1}^{\infty} c^i = \frac{c^0}{1-c} - \frac{c^{h+1}}{1-c} = \frac{c^{h+1} - 1}{c - 1} \quad (2.1.5)$$

Come abbiamo visto è possibile trovare formule chiuse per M_i, $|T_i|$, R_i e $C_{c,i-1}$, calcolabili in tempo costante. Sostituendo queste nell'Equazione 2.1.2 è possibile calcolare il valore $E(i)$ in tempo costante. L'algoritmo dunque cerca esaustivamente il livello i della frontiera, e calcola ogni volta il costo totale dell'albero con l'Equazione 2.1.2.

2.1.2.1 Pseudocodice

Mostriamo dunque lo pseudocodice dell'algoritmo.

Algoritmo 3: Modello 1, non limitato, albero regolare completo

```
1 E = +∞;
2 for i = 0 to h do
3 |   E(i) = c^i M_i r_d s_d + c^i (|T_i| − 1) r_q s_d α + R_i r_q s_d α + ((C_{c,i−1} e_tr + (C_{c,i} − 1) e_re) / (e_tr + e_re)) r_q s_q;
4 |   if E(i) < E then
5 |   |   E = E(i);
6 |   end
7 end
```

Per ricavare la colorazione ottima è sufficiente tenere traccia della profondità i della frontiera che miminizza E. L'algoritmo contiene un ciclo eseguito h volte, e ogni iterazione impiega tempo costante, dunque l'algoritmo ha complessità $O(h) = O(\lfloor \log_c n \rfloor)$.

2.1.3 Cammino

Assumiamo ora che l'albero in ingresso sia un cammino composta da n nodi, numerati in ordine, dove 0 è la radice, ed $n-1$ l'unica foglia. Applicando il Teorema 2.1.1 degli antenati al nostro caso, si capisce che la soluzione ottima sarà composta da un insieme di nodi neri consecutivi, seguiti da soli nodi bianchi. Assumiamo che in una soluzione ci siano k nodi neri, e dunque l'ultimo nodo nero sia $k-1$. La Figura 2.1.4 mostra un esempio di soluzione ottima. Poiché ogni nodo ha esattamente un figlio, il costo di invio e ricezione dell'interrogazione come

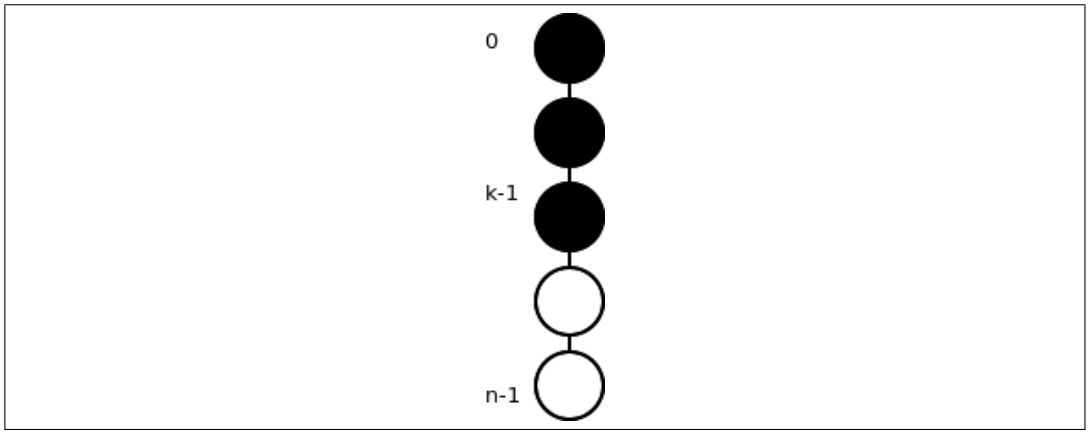


Figura 2.1.4: Esempio di soluzione ottima su cammino non limitato con $k = 3, n = 5$.

definito nell'Equazione 1.1.1, con $v_i = 1$ diviene

$$b_i = 1, \forall i$$

ignoreremo dunque tutti i costi energetici.

Sia $C(k)$ il costo totale dell'albero quando $k-1$ è l'ultimo nodo nero ed indichiamo con $Path(i, j)$ il cammino tra i nodi i e j. Possiamo calcolare $C(k)$ sommando i costi:

- $\sum_{i=0}^{k-1}(ir_q s_d \alpha)$: risalita delle informazioni generate dai nodi neri in $Path(0, k-1)$,
- $(k-1)r_q s_q$: costo di diffusione dell'interrogazione,
- $\sum_{i=k}^{n-1}[(i-(k-1))r_d s_d]$: messaggi grezzi scambiati in $Path(k, n-1)$,
- $(k-1)(n-k)r_q s_d \alpha$: risalita da $k-1$ a 0 delle informazioni generate da $Path(k, n-1)$.

Riassumendo otteniamo

$$\begin{aligned}
C(k) =& \\
=&\sum_{i=0}^{k-1}(ir_qs_d\alpha) + (k-1)r_qs_q + \sum_{i=k}^{n-1}[(i-(k-1))r_ds_d] + (k-1)(n-k)r_qs_d\alpha \\
=&\frac{(k-1)k}{2}r_qs_d\alpha + (k-1)r_qs_q + \sum_{i=1}^{n-1-(k-1)}[(i+(k-1)-(k-1))r_ds_d] \\
&+ (k-1)(n-k)r_qs_d\alpha \\
=&\frac{(k-1)k}{2}r_qs_d\alpha + (k-1)r_qs_q + \sum_{i=1}^{n-k} ir_ds_d + (k-1)(n-k)r_qs_d\alpha \\
=&\frac{(k-1)k}{2}r_qs_d\alpha + (k-1)r_qs_q + \frac{(n-k)(n-k+1)}{2}r_ds_d + (k-1)(n-k)r_qs_d\alpha \\
=&\frac{(n-k)(n-k+1)}{2}r_ds_d + \alpha\left[\frac{(k-1)k}{2} + (k-1)(n-k)\right]r_qs_d + (k-1)r_qs_q
\end{aligned} \tag{2.1.6}$$

È dunque possibile calcolare il costo $C(k)$ in tempo costante. Un primo algoritmo, simile a quello esposto nella Sezione 2.1.2, può cercare in modo esaustivo l'ultimo nodo nero $k-1$, calcolando $C(k)$ ad ogni iterazione. Un agoritmo siffatto avrebbe costo $O(n)$. Noi cercheremo di fare meglio, dimostrando che vale la proprietà bitonica:

Teorema 2.1.7. *Il cammino gode di* ***proprietà bitonica****, cioè facendo scendere passo passo dalla radice verso la foglia l'ultimo nero $k-1$, il costo dell'albero $C(k)$ prima descresce e poi cresce.*

Dimostrazione. Per verificare che ciò sia vero, riscriviamo l'Equazione 2.1.6 come

$$C(k) = \sum_{i=0}^{k-1}(ir_qs_d\alpha) + (k-1)r_qs_q + \sum_{i=1}^{n-k} ir_ds_d + (k-1)(n-k)r_qs_d\alpha$$

e calcoliamo poi $\triangle C_{k+1} = C(k+1) - C(k)$

$$\begin{aligned}
\triangle C_{k+1} =& C(k+1) - C(k) = \\
=&+\sum_{i=0}^{k}(ir_qs_d\alpha) - \sum_{i=0}^{k-1}(ir_qs_d\alpha) \\
&+ (k)r_qs_q - (k-1)r_qs_q \\
&+ \sum_{i=1}^{n-k-1} ir_ds_d - \sum_{i=1}^{n-k} ir_ds_d \\
&+ (k)(n-k-1)r_qs_d\alpha - (k-1)(n-k)r_qs_d\alpha \\
=&kr_qs_d\alpha + r_qs_q - (n-k)r_ds_d + (kn-k^2-k-kn+k^2+n-k)r_qs_d\alpha \\
=&kr_qs_d\alpha + r_qs_q - (n-k)r_ds_d + (-2k+n)r_qs_d\alpha \\
=&(k-n)(r_ds_d - r_qs_d\alpha) + r_qs_q
\end{aligned} \tag{2.1.7}$$

Possiamo facilmente notare che tale funzione è crescente al crescere di k. Dunque

$$C(k+1) - C(k) > 0 \Rightarrow C(k+2) - C(k+1) > 0$$

Questo significa che quando $C(k+1)$ è maggiore di $C(k)$, da quel momento in poi il costo continuerà a crescere con l'aumentare di k. Perciò il costo $C(k)$, all'aumentare di k, può prima scendere e poi crescere, senza altre variazioni.[2] □

Questa proprietà suggerisce che possiamo migliorare l'algoritmo, effettuando una ricerca binaria del minimo $C(k)$ in una sequenza bitonica di n elementi. L'algoritmo avrebbe quindi costo $O(\log n)$.

È possibile però fare di meglio. Osserviamo che l'Equazione 2.1.7 è crescente in k, ed è anche possibile determinare per quale k si ottiene $\triangle C_{k+1} \geq 0$. Impostando l'equazione $\triangle C_{k+1} = 0$ otteniamo che $k' = \left\lfloor n - \frac{r_q s_q}{r_d s_d - \alpha r_q s_d} \right\rfloor$ è l'ultimo valore per cui $\triangle C_{k+1} \leq 0$, ovvero $C(k'+1) - C(k') \leq 0$ e $C(k'+2) - C(k'+1) \geq 0$, cioè $C(k'+1) \leq C(k')$ e $C(k'+2) \geq C(k'+1)$, dunque $C(k)$ ha un minimo in corrispondenza di $k'+1$. Ne segue che per ottimizzare il costo del cammino, dobbiamo prendere come ultimo nodo nero quello che ha indice

$$k' = \left\lfloor n - \frac{r_q s_q}{r_d s_d - \alpha r_q s_d} \right\rfloor$$

Possiamo quindi calcolare l'ultimo nodo nero in tempo $O(1)$, e sarà poi sufficiente rendere neri tutti i nodi che lo precedono. Notiamo che essendo k' l'ultimo nodo nero, l'albero conterrà nella soluzione ottima $k'+1$ nodi neri.

Per completezza mostriamo lo pseudocodice dell'algoritmo

2.1.3.1 Pseudocodice

Algoritmo 4: Modello 1, non limitato, cammino

```
1 k' = ⌊ n − r_q s_q / (r_d s_d − α r_q s_d) ⌋;
2 for i = 0 to k' do
3 |   Poni i nero;
4 end
```

Analizziamone ora il costo; se siamo interessati solo a capire quale sia la soluzione ottima, possiamo limitarci a calcolare k', in tempo costante. Se invece vogliamo creare la soluzione ottima, dobbiamo rendere neri i primi $k'+1$ nodi, e questo si effettua in tempo $O(k')$. Se poi vogliamo anche conoscere il costo della soluzione ottima, allora lo otteniamo calcolando in tempo costante $C(k'+1)$.

[2] Può anche essere solo crescente, dunque in questo caso il costo salirà con k, e l'ottimo si ottiene con $k-1=0$. Il costo $C(k)$, all'aumentare di k, può inoltre essere solo decrescente, in questo caso l'ottimo si avrà con solo nodi neri.

2.1.4 Definizione caterpillar

Definiamo per primo cosa sia un grafo caterpillar, per poi definire gli algoritmi in grado di trattare tali grafi. Informalmente, il caterpillar è un albero, dove si distingue un cammino centrale composto da $h+1$ nodi, detto **dorsale**, di cui la radice è un estremo. Ogni nodo i della dorsale può avere inoltre un qualsiasi numero di figli foglia $\boldsymbol{F(i)}$. Vediamo un esempio di caterpillar in Figura 2.1.5. Per diverse definizioni di caterpillar ad altre informazioni si veda [12]. Nei caterpillar indicizziamo solo i nodi della dorsale, a partire da 0, la radice, fino ad $\boldsymbol{h}$, l'ultimo nodo della dorsale. Si noti che h, può anch'esso avere foglie ma alcuni algoritmi richiederanno che non ne abbia: in tal caso è sufficiente includere una foglia di h nella dorsale[3]. Data questa indicizzazione, il nodo x si trova a profondità x, e le sue foglie a profondità $x+1$. Nel presentare gli algoritmi assumeremo di sapere che l'albero di input sia un caterpillar, e di conoscere tutti i valori $F(i), \forall i \in [0,h]$.

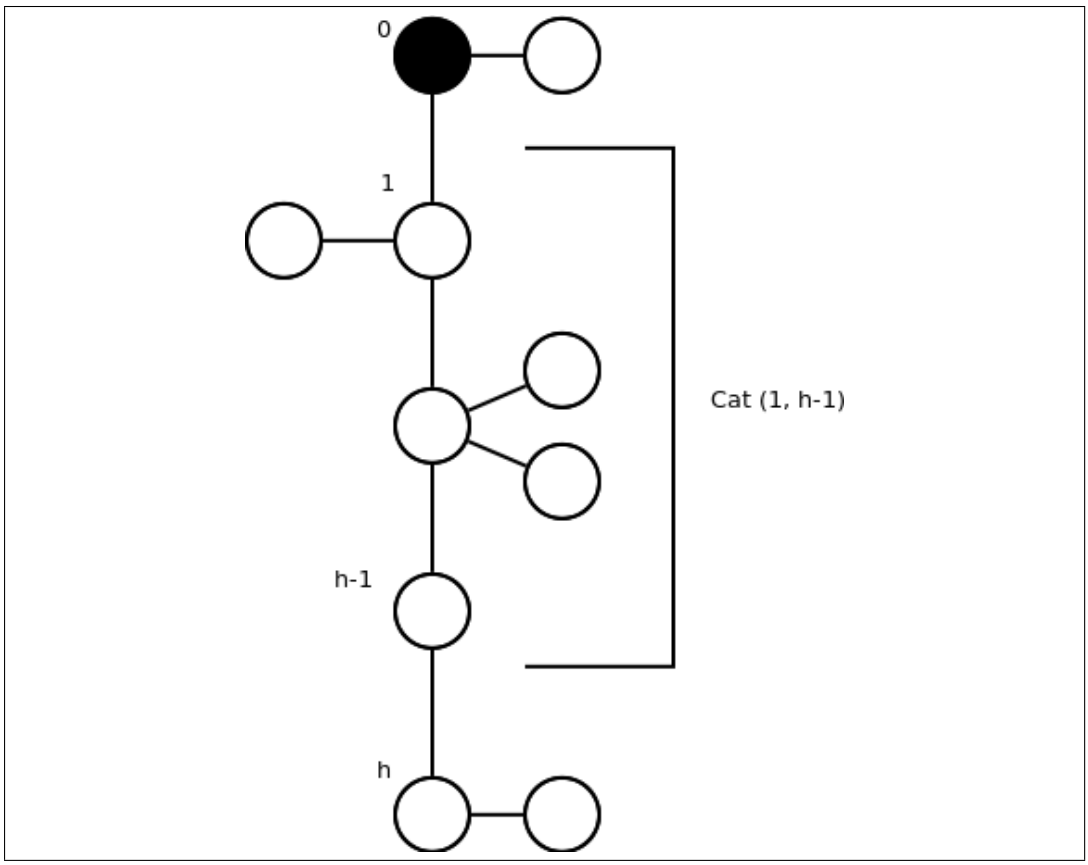


Figura 2.1.5: Esempio di caterpillar con radice nera.

Definiamo inoltre

Definizione 2.1.8.
$\boldsymbol{Cat(a,b)}$= sottografo del caterpillar originale che contiene tutti i nodi compresi tra a e b e le loro foglie, includendo anche le foglie di a e di b, ed escludendo l'eventuale figlio non foglia di b; è anch'esso un caterpillar. Con $Cat(0,h)$ si indica tutto il caterpillar di input.

[3]Tale foglie diventerà il nuovo nodo h.

Durante la trattazione di grafi caterpillar, useremo anche altre definizioni aggiuntive che utilizzeremo negli algoritmi dedicati. Siano dunque

$\boldsymbol{N_i} =$ Numero di nodi in $Cat(i, h)$; identico a $|T_i|$

$\boldsymbol{N'_i} =$ Numero di nodi in $Cat(0, i)$.

$\boldsymbol{M_i} =$ Numero di messaggi grezzi e di risposta scambiati in $Cat(i, h)$, ovvero la risalita fino ad i delle informazioni generate dai nodi in $Cat(i, h)$. M_0 è il numero totale di messaggi grezzi e di risposta in tutto il caterpillar.

$\boldsymbol{M'_i} =$ Numero di messaggi grezzi e di risposta scambiati in $Cat(0, i)$, ovvero la risalita delle informazioni generate dai nodi in $Cat(0, i)$ fino a 0. Non comprende la risalita fino alla radice delle informazioni generate da $Cat(i + 1, h)$.

$\boldsymbol{F(i, j)} =$ Numero di foglie in $Cat(i, j)$.

Si noti che valgono le seguenti relazioni:

$$\begin{aligned} N_i &= N_{i+1} + 1 + F(i), && \forall i \in [0, h] \\ N'_i &= n - N_{i+1}, && \forall i \in [0, h] \\ M_i &= M_{i+1} + N_{i+1} + F(i), && \forall i \in [0, h] \\ M'_i &= M_0 - M_{i+1} - N_{i+1}(i + 1), && \forall i \in [0, h] \\ F(i, j) &= M'_j - M'_i - [N'_j - N'i]i \end{aligned}$$

Da cui possiamo calcolare i valori N_i, N'_i, M_i, M'_i, tramite gli Algoritmi 5, 6, 7 e 8 di complessità $O(h)$.

Vale inoltre la seguente relazione

$$M_0 = M'_i + M_{i+1} + N_{i+1}(i + 1), \forall i \in [0, h] \tag{2.1.8}$$

Algoritmo 5: Caterpillar N_i

```
1 foreach nodo i in dorsale in postordine do
2 |   if i = h then
3 |   |   N_i = F(h) + 1;
4 |   else
5 |   |   N_i = N_{i+1} + 1 + F(i);
6 |   end
7 end
```

Algoritmo 6: Caterpillar N'_i

```
1 foreach nodo i in dorsale, in postordine do
2 |   if i = h then
3 |   |   N'_i = n;
4 |   else
5 |   |   N'_i = n − N_{i+1};
6 |   end
7 end
```

Algoritmo 7: Caterpillar M_i

```
1 foreach nodo i in dorsale, in postordine do
2 |   if i = h then
3 |   |   M_i = F(i);
4 |   else
5 |   |   M_i = M_{i+1} + N_{i+1} + F(i);
6 |   end
7 end
```

Algoritmo 8: Caterpillar M'_i

```
1 foreach nodo i in dorsale, in postordine do
2 |   if i = h then
3 |   |   M'_i = M_0
4 |   else
5 |   |   M'_i = M_0 − M_{i+1} − N_{i+1}(i + 1);
6 |   end
7 end
```

2.1.5 Caterpillar

Risolviamo ora il nostro problema nel caso non limitato quando il grafo in input è un caterpillar. In questo algoritmo supponiamo che il nodo h sia sempre una foglia. Ricordiamo il Teorema 2.1.1 degli antenati ed il Teorema 2.1.2 dei fratelli ed applichiamoli ad un albero caterpillar, ottenendo:

Teorema 2.1.9. *Si consideri un grafo caterpillar, nel caso non limitato e nel modello 1. Nel caso ottimo, la frontiera è composta da un unico nodo q appartenente alla dorsale. Tutti i suoi antenati ed i loro figli sono neri, mentre tutti gli altri nodi sono bianchi; quindi* $Cat(0, q-1)$ *contiene solo nodi neri,* $Cat(q+1, h)$ *contiene solo nodi bianchi, ed i figli di q sono tutti bianchi.*

Siamo infatti sicuri che tra i nodi neri a profondità massima ve ne sia sempre uno in backbone per il Teorema 2.1.2 dei fratelli; se infatti una foglia fosse nera, allora anche il fratello in dorsale dovrà essere nero.
La soluzione ottima dunque apparirà come in Figura 2.1.6.

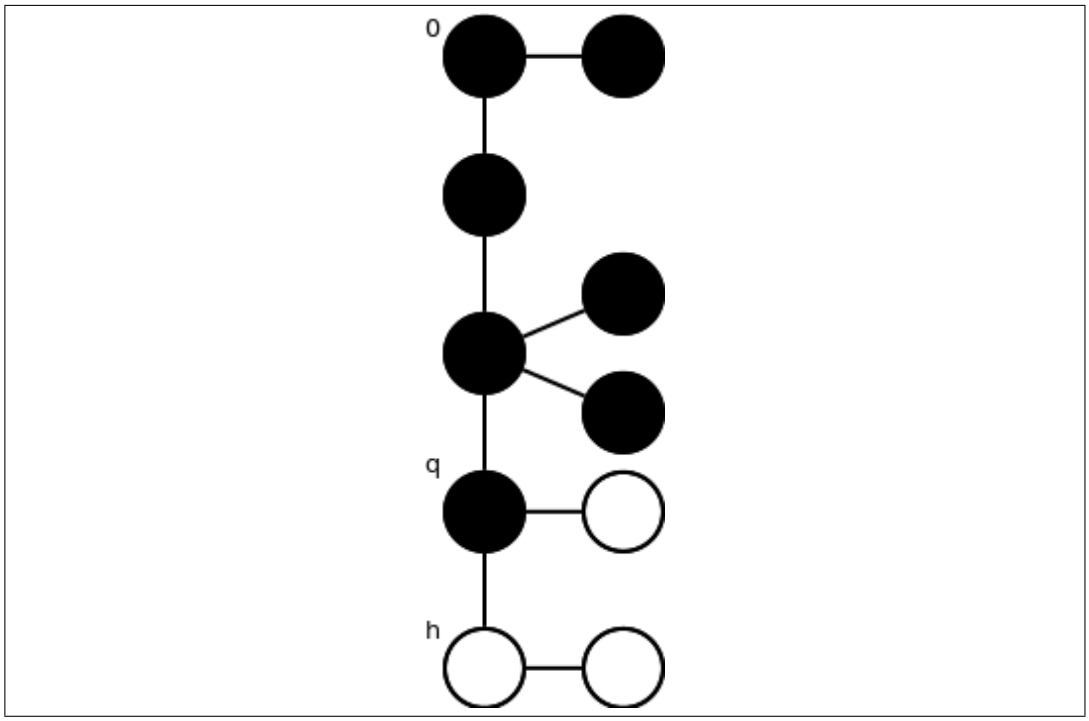


Figura 2.1.6: Esempio di soluzione ottima su caterpillar, caso non limitato, modello 1.

L'algoritmo si basa quindi sulla ricerca del nodo nero a profondità massima. La ricerca viene fatta esaustivamente su tutti gli $h+1$ nodi nella dorsale, e ad ogni iterazione viene calcolato il costo totale dell'albero. Definiamo il costo come segue:

Definizione 2.1.10. sia $E(q)$ il costo totale dell'albero dove l'ultimo nodo nero nella dorsale è q. $E(q)$ comprende i costi dei soli messaggi grezzi e di risposta.

Avendo $E(q)$ si può calcolare il costo totale E dell'albero tramite l'Equazione 2.1.9 :

$$E = \min_{\forall q, q \leq h} \left\{ E(q) + \frac{q e_{tr} + \left[N'_q - F(q) \right] e_{re}}{e_{tr} + e_{re}} r_q s_q \right\} \tag{2.1.9}$$

Questo è possibile poiché il costo totale per la trasmissione dell'interrogazione dipende solo dalla profondita di q. Dato che q è l'ultimo nero, l'interrogazione deve essere inviata da tutti i suoi q antenati, e deve essere ricevuta da tutti i N'_q nodi in $Cat(0, q)$ meno che dalle foglie di q.

È possibile ora calcolare $E(q)$ sommando i costi:

- $M_q r_d s_d$: i messaggi grezzi scambiati in $Cat(q, h)$,
- $|T_q| q r_q s_d \alpha$: la risalita da q a 0 delle informazioni generate in $Cat(q, h)$, sottoforma di messaggi di risposta,
- $M'_{q-1} r_q s_d \alpha$: i messaggi di risposta scambiati in $Cat(0, q-1)$.

Abbiamo quindi

$$E(q) = |T_q| q r_q s_d \alpha + M_q r_d s_d + M'_{q-1} r_q s_d \alpha \tag{2.1.10}$$

Ricordando che $|T_q| = N_q$, i valori $N_i, N'_i, M_i, M'_i, \forall i \in [0, h]$ saranno calcolati in precomputazione mediante gli Algoritmi 5,6,7 e 8 di complessità $O(h)$; tali algoritmi rappresentano tutte e sole le precomputazioni da effettuare. Avendo a disposizione tali valori è possibile calcolare $E(q)$ in tempo costante.

2.1.5.1 Pseudocodice

Mostriamo ora lo pseudocodice dell'algoritmo:

Algoritmo 9: Modello 1, non limitato, caterpillar

```
1  Effettua precomputazioni;
2  foreach q ∈ [0, h] do
3  |  E(q) = |T_q| q r_q s_d α + M_q r_d s_d + M'_{q-1} r_q s_d α;
4  end
5  E = +∞;
6  foreach q ∈ [0, h] do
7  |  E_temp = { E(q) + ((q e_tr + [N'_q − F(q)] e_re) / (e_tr + e_re)) r_q s_q };
8  |  if E_temp < E then
9  |  |  E = E_temp;
10 |  end
11 end
```

L'algoritmo appartiene alla classe di complessità $O(h)$, ed al termine della computazione la variabile E contiene il minimo costo dell'albero. Per ricavare la colorazione ottima è sufficiente tenere traccia del nodo q che minimizza E. Per colorare effettivamente i nodi neri è invece necessario tempo $O(k_u)$.

2.1.5.2 Correttezza ed ottimalità

Una volta mostrata come deve apparire la soluzione ottima, si effettua una ricerca esaustiva di tutti i possibili ultimi nodi neri in dorsale. La verifica di correttezza dunque si deve

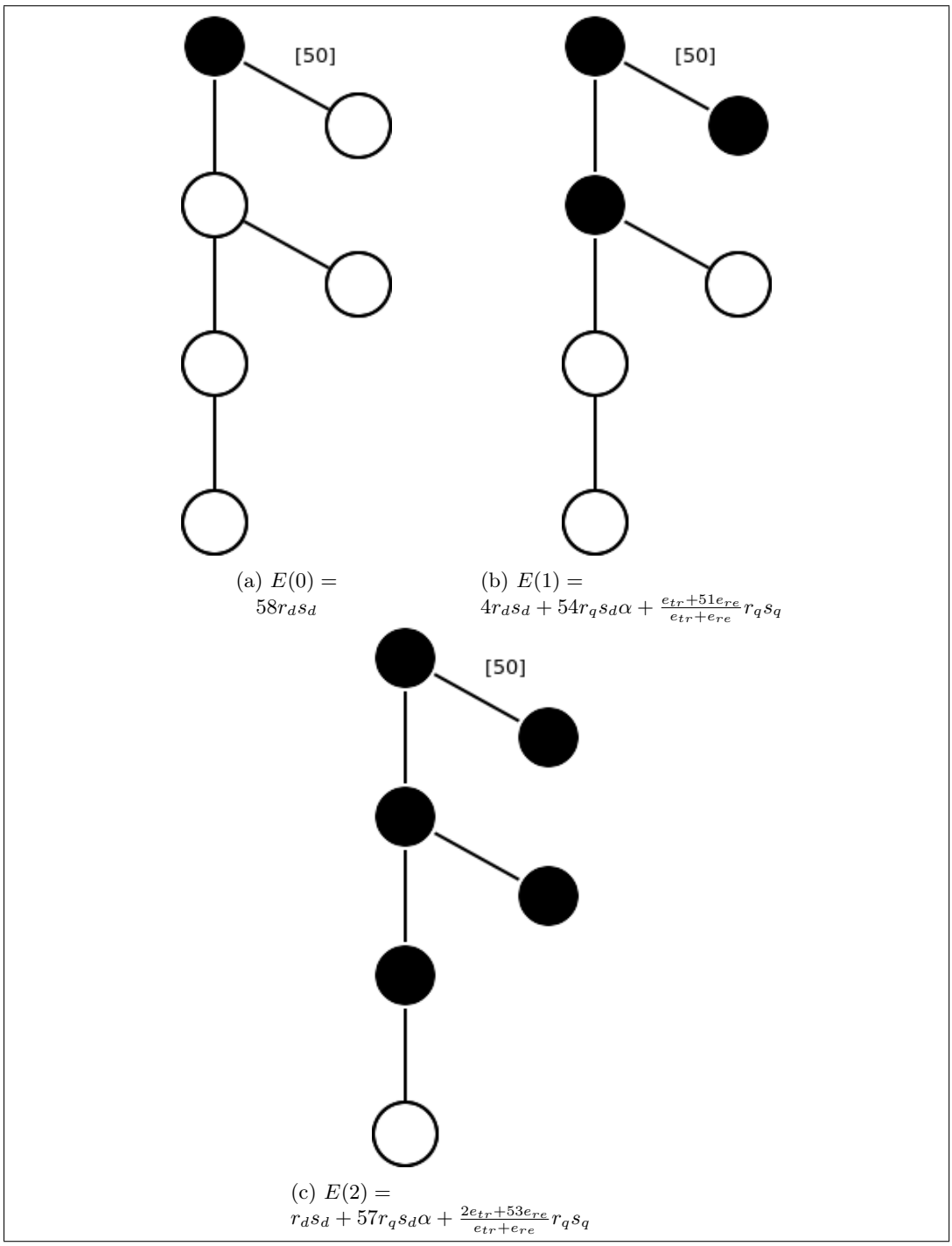
limitare a verificare che sia ben definito $E(q)$ nell'Equazione 2.1.10 e che sia corretto E nell'Equazione 2.1.9. La definizione di E è corretta poiché cerca esaustivamente il caso migliore tra tutti i possibili $E(q)$ aggiungendo poi il costo di invio dell'interrogazione.
Per convincersi che la definizione di $E(q)$ è corretta procediamo verificando che essa conteggi il numero esatto (M_0) di messaggi scambiati. Dall'Equazione 2.1.10 si evince che vengono inviati in tutto $|T_q|q + M_q + M'_{q-1}$ messaggi, e riscrivendo otteniamo $N_q q + M_q + M'_{q-1}$. Ora, se dividiamo idealmente l'albero di input in $Cat(0, q-1)$ e $Cat(q, h)$, possiamo ottenere il numero di messaggi M_0 sommando i messaggi scambiati in $Cat(0, q-1)$, sommando i messaggi scambiati in $Cat(q, h)$ e sommando la risalita dei messaggi generati da $Cat(q, h)$, che vanno da q a 0. Tale quantità è proprio $M_0 = M'_{q-1} + M_q + N_q q$. Di questi solo quelli scambiati in $Cat(q, h)$, cioè M_q, sono messaggi grezzi, mente gli altri sono messaggi di risposta.

Ottimizzazione

Possiamo provare ad ottimizzare parte dell'algoritmo. Ci chiediamo se valga la proprietà bitonica per un caterpillar generico:

Definizione 2.1.11. Un grafo caterpillar gode di **proprietà bitonica**, se è vera l'affermazione: *facendo scendere, dalla radice verso le foglie, la profondità q dell'ultimo nodo nero, il costo $E(q)$ prima decresce e poi cresce.*

Se ciò fosse vero sarebbe possibile effettuare una ricerca binaria per individuare il minimo in una sequenza bitonica in tempo $O(\log h)$ (ma non abbatterebbe il costo delle precomputazioni $O(h)$). Come vediamo però nel controesempio in Figura 2.1.7 tale proprietà non vale. Per dimostrare che non vale la bitonicità, dimostriamo che esiste un particolare caterpillar, e particolari valori delle variabili, per cui $E(q-1) < E(q) > E(q+1)$.



(a) $E(0) = 58 r_d s_d$

(b) $E(1) = 4 r_d s_d + 54 r_q s_d \alpha + \frac{e_{tr}+51 e_{re}}{e_{tr}+e_{re}} r_q s_q$

(c) $E(2) = r_d s_d + 57 r_q s_d \alpha + \frac{2e_{tr}+53 e_{re}}{e_{tr}+e_{re}} r_q s_q$

Figura 2.1.7: Controesempio che mostra come non valga la proprietà bitonica. Il simbolo *[50]* indica che la radice ha 50 foglie di quel tipo. Imponendo $r_q = s_q = s_d = 1, \alpha = 1/2, e_{re} = 1, e_{tr} = 1/2, r_d = 1.1$ si verifica che $E(0) < E(1) > E(2)$, e sono soddisfatti i vincoli $\alpha r_q < r_d$. In particolare
$E(0) = 58 * 1.1 = 63.8$,
$E(1) = 4 * 1.1 + 54/2 + \frac{0.5+51}{0.5+1} = 65.7\overline{3}$,
$E(2) = 1.1 + 57/2 + \frac{2/2+53}{0.5+1} = 65.6$
da cui $63.8 < 65.7\overline{3} > 65.6$.

2.1.6 Caterpillar regolare

Assumiamo ora un caso molto particolare, dove l'albero in input è un caterpillar regolare. In un caterpillar regolare ogni nodo della dorsale ha un numero costante c di figli, ad eccezione del nodo h che sarà sempre una foglia. Ogni nodo $i, i \in [0, h-2]$ avrà k figli foglia (con $c = k+1$), il nodo $h-1$ avrà invece c figli foglia. Il teorema 2.1.9 a pagina 24 è applicabile a questo caso, dunque lnella soluzione ottima la frontiera è costituita da un unico nodo x nella dorsale. Vediamo un esempio di caterpillare regolare in Figura 2.1.8 con una delle possibili colorazioni ottime. Sui caterpillar regolari valgono ovviamente le proprietà descritte nella

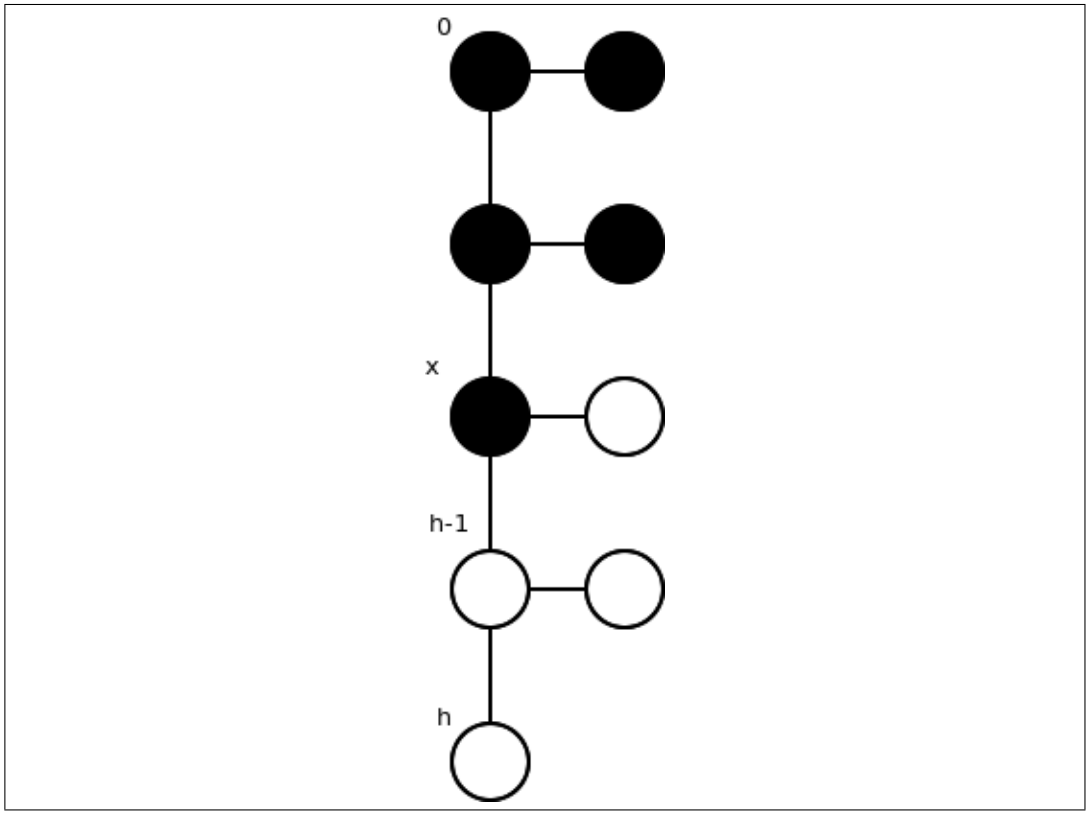


Figura 2.1.8: Esempio di caterpillar regolare con $c = 2, k = 1$.

Sezione 2.1.5, ed in particolare vale il Teorema 2.1.9, che ci dice che la frontiera è composta da un solo nodo x in dorsale, i suoi fratelli sono neri, e tutti i suoi discendenti bianchi.
Poiché ogni nodo ha c figli, possiamo riscrivere l'Equazione 1.1.1 di costo per l'invio di messaggi ai figli come

$$b_i = \frac{e_{tr} + ce_{re}}{e_{tr} + e_{re}}$$

ovvero $b_i = b_0, \forall i \in [0, h-1]$ e $b_h = 0$.

Per studiare questo caso definiamo la seguente nozione di costo:

Definizione 2.1.12. $C(x)$ = costo dell'albero quando l'ultimo nero è x, tutti i suoi predecessori sono neri, e tutti i suoi discendenti sono bianchi. Non comprende il costo di uscita delle informazioni dalal radice verso il padre fittizio.

Possiamo calcolare $C(x)$ sommando i seguenti costi

- $\sum_{i=1}^{x} i(k+1)\alpha r_q s_d$: messaggi che trasportano informazioni generate dai nodi neri (compreso x),

- $\sum_{i=1}^{h-x} i(k+1) r_d s_d$: messaggi grezzi scambiati in $Cat(0,h)$,

- $\sum_{i=1}^{x} b_i r_q s_q$: invio e ricezione dell'interrogazione,

- $(h-x)(k+1)x\alpha r_q s_d$: risalita da x a 0 delle informazioni generate dai nodi bianchi.

Ricomponendo abbiamo:

$$C(x) = \sum_{i=1}^{x} i(k+1)\alpha r_q s_d + \sum_{i=1}^{h-x} i(k+1) r_d s_d + \sum_{i=1}^{x} b_i r_q s_q + (h-x)(k+1)x\alpha r_q s_d \quad (2.1.11)$$

La prima idea per risolvere il problema è di cercare esaustivamente il valore $\min_{x\in[0,h]}\{C(x)\}$, tra tutti i possibili x, ma cerchiamo invece un ottimizzazione per non dover effettuare tale ricerca. Studiamo quindi la funzione $C(x+1) - C(x)$:

$$
\begin{aligned}
&\triangle C(x+1) = C(x+1) - C(x) = \\
=&\sum_{i=1}^{x+1} i(k+1)\alpha r_q s_d - \sum_{i=1}^{x} i(k+1)\alpha r_q s_d \\
&+ \sum_{i=1}^{h-x-1} i(k+1) r_d s_d - \sum_{i=1}^{h-x} i(k+1) r_d s_d \\
&+ \sum_{i=1}^{x+1} b_i r_q s_q - \sum_{i=1}^{x} b_i r_q s_q \\
&+ (h-x-1)(k+1)(x+1)\alpha r_q s_d - (h-x)(k+1)x\alpha r_q s_d = \\
\\
=&(x+1)(k+1)\alpha r_q s_d \\
&- (h-x)(k+1) r_d s_d \\
&+ b_x r_q s_q \\
&+ (h-x)(k+1)(x+1)\alpha r_q s_d - (k+1)(x+1)\alpha r_q s_d - (h-x)(k+1)x\alpha r_q s_d = \\
\\
=&x(k+1)\alpha r_q s_d + (k+1)\alpha r_q s_d \\
&- h(k+1) r_d s_d + x(k+1) r_d s_d \\
&+ b_x r_q s_q + \\
&+ (h-x)(k+1)\alpha r_q s_d - (k+1)(x+1)\alpha r_q s_d = \\
\\
=&x(k+1)\alpha r_q s_d + (k+1)\alpha r_q s_d \\
&- h(k+1) r_d s_d + x(k+1) r_d s_d \\
&+ b_x r_q s_q \\
&+ (k+1)\alpha r_q s_d [h - 2x - 1] = \\
\\
=&(k+1)\alpha r_q s_d \\
&- h(k+1) r_d s_d + x(k+1) r_d s_d \\
&+ b_x r_q s_q \\
&- x(k+1)\alpha r_q s_d + (h-1)(k+1)\alpha r_q s_d = \\
\\
=&x(k+1)\left(r_d s_d - \alpha r_q s_d\right) - h(k+1) r_d s_d + h(k+1)\alpha r_q s_d + b_x r_q s_q
\end{aligned}
\tag{2.1.12}
$$

Da cui si può vedere che è crescente al crescere di x, poiché $\alpha r_q < r_d$. Questa funzione è prima negativa e poi positiva, ed indica che per $\forall x, C(x+2) - C(x+1) \geq C(x+1) - C(x)$, ovvero la funzione $C(x)$ è bitonica, prima decrescente e poi decrescente. Possiamo dire dunque che il caterpillar regolare gode di **proprietà bitonica** come espressa nella Definizione 2.1.11. Questo suggerische che per cercare il nodo x che minimizza $C(x)$ possiamo effettuare una ricerca binaria per individuare il minimo su una sequenza bitonica in tempo $O(\log h)$.

Si può però fare di meglio cercando l'ultimo valore di x per cui $C(x+1) - C(x) \leq 0$, ed esso si ottiene impostando l'equazione $\triangle C(x+1) = 0$, ottenendo così:

$$x' = \left\lfloor \frac{h(k+1)r_d s_d - h(k+1)\alpha r_q s_d - b_i r_q s_q}{(r_d s_d - \alpha r_q s_d)(k+1)} \right\rfloor \tag{2.1.13}$$

Dunque $C(x'+1) - C(x') \leq 0$ e $C(x'+2) - C(x'+1) \geq 0$, quindi la funzione $C(x)$ avrà il minimo nel punto $x = x' + 1$, ed esso è l'indice dell'ultimo nodo nero .

2.1.6.1 Pseudocodice

Lo pseudocodice dell'algoritmo è molto semplice:

Algoritmo 10: Modello 1, non limitato, caterpillar regolare

1 $x' = \left\lfloor \frac{h(k+1)r_d s_d - h(k+1)\alpha r_q s_d - b_i r_q s_q}{(r_d s_d - \alpha r_q s_d)(k+1)} \right\rfloor$;
2 Poni nero ogni nodo con profondità al più $x' + 1$;

Si noti che è possibile calcolare x' in tempo costante. Se vogliamo conoscere solamente quale sia l'ultimo nodo nero, è sufficiente calcolare $x' + 1$ in tempo costante. Se vogliamo creare la colorazione ottima, dovremo rendere neri i nodi approppriati in tempo $O(k_u)$. Se infine ci interessa il costo della soluzione ottima dovremo calcolare $C(x'+1)$ in tempo costante tramite l'Equazione 2.1.14.

$$\begin{aligned} C(x) = & \\ &= \sum_{i=1}^{x} i(k+1)\alpha r_q s_d + \sum_{i=1}^{h-x} i(k+1) r_d s_d + \sum_{i=1}^{x} b_i r_q s_q + (h-x)(k+1)x\alpha r_q s_d \\ &= \frac{x(x+1)}{2}(k+1)\alpha r_q s_d + \frac{(h-x)(h-x+1)}{2}(k+1) r_d s_d + \\ &\quad + x b_i r_q s_q + (h-x)(k+1)x\alpha r_q s_d \end{aligned} \tag{2.1.14}$$

2.2 Limitato

Nella sezione precedente abbiamo supposto di avere a disposizione un qualsiasi numeri di nodi neri, chiamando tale problema *caso non limitato*; nella realtà invece le memorie da applicare ai sensori hanno un costo hardware, dunque ne potremo utilizzare solo un numero limitato definito a priori, chiamiamo tale problema **caso limitato**; negli algoritmi che seguono supponiamo di avere a disposizione esattamente $\boldsymbol{k}$ nodi neri da piazzare, di questi uno sarà sempre la radice, mentre gli altri possono essere utilizzati tutti o in parte; poiché la radice sarà sempre nera supporremo che $k \geq 1$. Visto che il numero di nodi neri è limitato, il problema diventa più complesso, fondamentalmente la disposizione dei nodi neri è un tradeoff tra due casi: ponendo i nodi con memoria vicino alla radice un maggior numero di dati viene memorizzato e dunque trasferito come messaggio di risposta, il costo per la diffusione dell'interrogazione sarà minore, ma verrà utilizzata più energia per trasferire i dati grezzi attraverso una porzione di albero molto più lunga; se invece poniamo le memorie nei nodi lontani dalla radice i dati grezzi attraverseranno lunghezze minori, ma un numero minore di dati verrà trasferito come messaggio di risposta ed il costo di diffusione dell'interrogazione aumenterà.

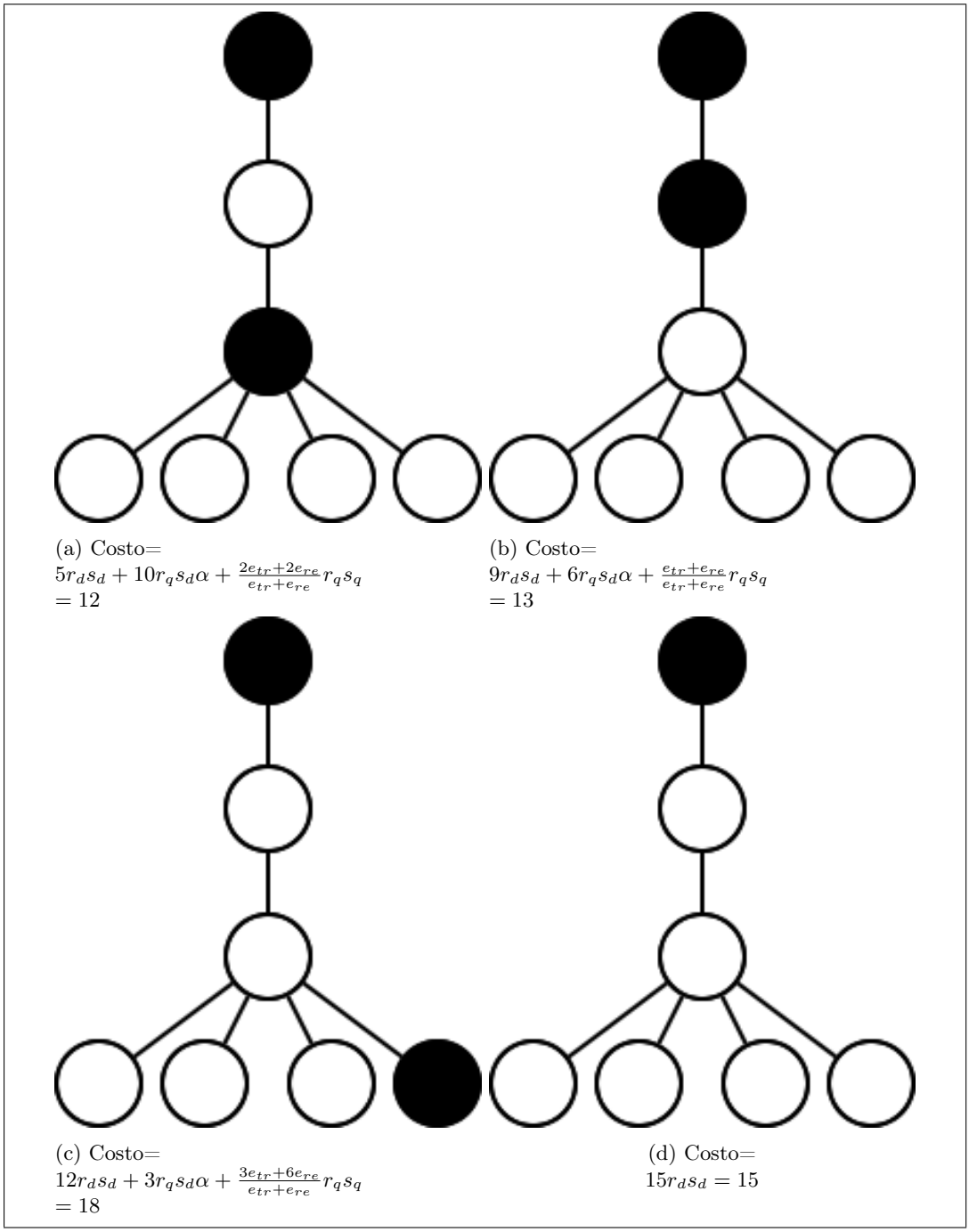
Il costo energetico di invio delle informazioni definito dall'Equazione 1.1.1 non varia nel caso limitato, poiché dipende solo dal modello di comunicazione utilizzato.

Il Teorema 2.1.2 dei fratelli visto nel caso non limitato non vale più, poiché potremmo non avere un numero sufficiente di sensori con memoria, possono cioè esistere nodi fratelli di colore diverso. Il Teorema 2.1.1 degli antenati non vale, è infatti semplice verificare con un esempio che nel caso ottimo un nodo nero può avere antenati bianchi; un esempio è presentato in Figura 2.2.1.

Possiamo comunque dire che vale ancora il Teorema 1.1.3 se abbiamo un numero sufficiente di neri da piazzare.

Quando dobbiamo risolvere il problema limitato con k nodi su un certo albero di input, è opportuno risolvere prima il problema non limitato; questo ci dirà che la soluzione ottima con un qualsiasi numero di nodi neri si avrà utilizzando k_u neri. Quando $k \geq k_u$ la soluzione ottima nel caso limitato coinciderà con quella ottenuta nel caso non limitato. Poiché generalmente gli algoritmi presentati nel caso non limitato hanno costo inferiore o uguali a quelli nei rispettivi casi limitati tale consiglio è applicabile senza costi aggiuntivi. Per il momento supporremo che il costo degli algoritmi limitati sia sempre inferiore a quello degli algoritmi non limitati, proponendoci di verificarla nella conclusioni della tesi,

Per tali motivi gli algoritmi che presentiamo di seguito assumeranno sempre che $k < k_u$.



(a) Costo= $5r_d s_d + 10r_q s_d \alpha + \frac{2e_{tr}+2e_{re}}{e_{tr}+e_{re}} r_q s_q = 12$

(b) Costo= $9r_d s_d + 6r_q s_d \alpha + \frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} r_q s_q = 13$

(c) Costo= $12r_d s_d + 3r_q s_d \alpha + \frac{3e_{tr}+6e_{re}}{e_{tr}+e_{re}} r_q s_q = 18$

(d) Costo= $15r_d s_d = 15$

Figura 2.2.1: Controesempio che mostra come nel caso limitato nel primo modello un nodo nero possa avere antenati bianchi. Per $k = 2, r_d = s_d = r_q = s_q = e_{tr} = e_{re} = 1, \alpha = 0.5$ abbiamo verificato esaustivamente che la colorazione (a) è ottima. La colorazione (c) è identica a quelle ottenibili colorando una diversa foglia.

2.2.1 Albero

Risolviamo innanzitutto il problema nel caso più generico, in cui l'albero in input non abbia nessuna topologia specifica. L'algoritmo che presentiamo è stato originariamente proposto da *Bo Sheng et al.* [7]. Diversamente dagli altri algoritmi, qui indicizziamo i nodi in post ordine, dove n è la radice e d_i la profondità del nodo i. Nell'Equazione 1.1.2 abbiamo definito il costo ottimo $E(i)$ di un sottoalbero T_i come $E(i) = \sum_{i \in T_i} e(i)$; ora vogliamo cambiare questa definizione, in modo che comprenda anche il costo per far giungere le informazioni generate da tutti i nodi in T_i fino alla radice 0 dell'albero. Per calcolare il costo con la nuova definizione, procediamo dalle foglie verso la radice in postordine; $E(i)$ si calcola sommando i costi dei sottoalberi $\sum_{j \in C_i} E(j)$, e sommando il costo del nodo i, come definito in 2.2.1; chiameremo il costo di un sottoalbero $E_i(m, l)$ invece di $E(i)$.

Sia l il numero di nodi bianchi tra il nodo i ed il suo antenato nero più vicino; precisiamo che $0 \leq l \leq n-2$.

Ridefiniamo quindi il costo di un nodo come segue:

Definizione 2.2.1. Sia $e(i)$ il costo di un nodo i, definito come l'energia necessaria per far giungere l'informazione generata dal nodo i fino alla radice 0 dell'albero, utilizzando messaggi grezzi e/o di risposta.

Caso 1. se i è nero, allora il suo costo è $e(i) = d_i r_q s_d \alpha$, poiché questa informazione dovrà essere trasmessa esattamente d_i volte per giungere alla radice.

Caso 2. se i è bianco, il suo costo è $r_d s_d$ per ogni nodo (i compreso) che lo congiunge con il suo antenato nero più vicino, e $r_q s_d \alpha$ per ogni nodo dal suo antenato nero più vicino fino alla radice. Se un nodo i è bianco, allora $e(i) = (l+1) r_d s_d + (d_i - l - 1) r_q s_d \alpha$

riassumendo

$$e(i) = \begin{cases} d_i r_q s_d \alpha & \text{se } i \text{ nero} \\ (l+1) r_d s_d + (d_i - l - 1) r_q s_d \alpha & \text{altrimenti} \end{cases}$$

Con questa definizione possiamo poi calcolare il costo totale dell'albero tramite $E = \sum_i e(i)$ e sommando poi l'opportuno costo di discesa dell'interrogazione.

Sia inoltre m il numero massimo di nodi neri in T_i, esso inoltre è tale che $0 \leq m \leq k$. Definiamo allora il costo di un sottoalbero:

Definizione 2.2.2. Sia $E_i(l, m)$ il costo ottimo del sottoalbero T_i con al più m neri che include il costo di tutti i nodi di T_i come definito in 2.2.1, il costo di risalita delle informazioni fino alla radice 0 ed il costo di invio dell'interrogazione da i ai suoi discendenti. Se $m = 0$ nessun nodo nero è utilizzato nella colorazione di T_i, se invece $m \geq 1$, allora almeno un nodo nero è stato utilizzato nella colorazione di T_i.

Possiamo ridefinire per ricorrenza il costo $E_i(l, m)$. Per farlo, definiamo prima la seguente quantità, che corrisponde al costo per l'invio dell'interrogazione da parte di i quando T_i contiene m neri:

$$Q_0^i(m) = \begin{cases} 0 & \text{se } m = 0 \\ b_i r_q s_q & \text{se } m \geq 1 \end{cases}, Q_1^i(m) = Q_0^i(m-1) \tag{2.2.1}$$

definiamo quindi $E_i(l, m)$ nel modo seguente:

Definizione 2.2.3.

Caso 1. se i è una foglia, $E_i(l,m)$ include il costo di i ed costo per l'invio a tutti i suoi predecessori. In particolare se i è bianco il suo costo sarà $r_d s_d$, ed il costo che incorre nei suoi predecessori $l r_d s_d + (d_i - l) r_q \alpha s_d$, quindi

$$E_i(m,l) = \begin{cases} (l+1) r_d s_d + (d_i - l) r_q \alpha s_d & \text{se } m = 0 \\ (d_i + 1) r_q \alpha s_d & \text{se } m \geq 1 \end{cases} \tag{2.2.2}$$

Caso 2. se i è bianco non foglia, l'upper bound m deve essere diviso tra i $|C_i|$ figli. Sia dunque $P(m)$ l'insieme di tutte le permutazioni $p = \left(m_j^p | j \in C_i\right)$, dove $\sum_{j \in C_i} m_j^p = m$ e m_j^p denota il massimo numero di neri nel sottoalbero T_j nella permutazione p. $E_i(l,m)$ è dunque definito dalla somma dei costi di:

- $\min_{\forall p \in P(m)} \left\{ \sum_{j \in C_i} E_j \left(m_j^p, l+1\right) \right\}$: costo dei sottoalberi scelti in base alla permutazione che minimizza il loro costo totale.
- $r_d s_d + Q_0^i(m)$: costo di i.
- $l r_d s_d + (d_i - l) r_q \alpha s_d$: costo precalcolato per far giungere l'informazione di i alla radice.

Dunque riassumendo:

$$E_i(m,l) = \min_{\forall p \in P(m)} \left\{ \sum_{j \in C_i} E_j \left(m_j^p, l+1\right) \right\} + (l+1) r_d s_d + (d_i - l) r_q \alpha s_d + Q_0^i(m) \tag{2.2.3}$$

Caso 3. se i è nero non foglia, l'upper bound $m-1$ deve essere diviso tra i figli C_i. Sia dunque $P(m-1)$ l'insieme di tutte le permutazioni $p = \left(m_j^p | j \in C_i\right)$, dove $\sum_{j \in C_i} m_j^p = m - 1$ e m_j^p denota il massimo numero di neri nel sottoalbero T_j nella permutazione p. Dunque $E_i(m,l)$ si calcola sommano

- $\min_{\forall p \in (P(m-1))} \left\{ \sum_{j \in C_i} E_j \left(m_j^p, 0\right) \right\}$: costo dei sottoalberi scelti in base alla permutazione che minumizza il loro costo totale.
- $r_q \alpha s_d + Q_1^i(m)$: il costo di i.
- $d_i r_q \alpha s_d$: il costo precalcolato per far risalire l'informazione di i alla radice.

Dunque

$$E_i(m,l) = \min_{\forall p \in (P(m-1))} \left\{ \sum_{j \in C_i} E_j \left(m_j^p, 0\right) \right\} + (d_i + 1) r_q \alpha s_d + Q_1^i(m) \tag{2.2.4}$$

La Definizione 2.2.3 rappresenta la relazione di programmazione dinamica usata all'algoritmo.

2.2.1.1 Pseudocodice

L'algoritmo assume che i nodi siano numerati in postordine, e che sia conosciuta la loro profondità, entrambe queste informazione sono ottenibili in tempo $O(n)$. L'algoritmo mantiene una tabella $(k+1) \times (n-1)$ bi-dimensionale per ogni nodo i, chiamata $E_i[m,l]$, dove $0 \leq m \leq k$ e $0 \leq l \leq n-2$.

Algoritmo 11: Modello 1, limitato, albero generico

```
1  foreach nodo i do
2      for m = 0 to k − 1 do
3          for l = 0 to n − 2 do
4              if m = 0 then
5                  E_i[m,l] = (l+1) r_d s_d + (d_i − l) r_q α s_d;
6              if m ≥ 1 then
7                  E_i[m,l] = (d_i + 1) r_q α s_d
               end
           end
       end
   end
8  foreach nodo i non foglia, non radice, in postordine do
9      for m = 0 to k − 1 do
10         for l = 0 to n − 2 do
               // i bianco
11             min_1 = min_{∀p∈P(m)} {Σ_{j∈C_i} E_j[m_j^p, l+1]} + (l+1) r_d s_d + (d_i − l) r_q α s_d + Q_0^i(m);
               // i nero
12             min_2 = min_{∀p∈(P(m−1))} {Σ_{j∈C_i} E_j[m_j^p, 0]} + (d_i + 1) r_q α s_d + Q_1^i(m);
13             E_i[m,l] = min{min_1, min_2};
           end
       end
   end
14 E_n[k,0] = min_{∀p∈P(m−1)} { Σ_{j∈C_n} E_j [m_j^p, 0] } + r_q α s_d + Q_1^i(m);
15 E_w = Σ_{i≠radice} d_i r_d s_d;
16 E_n[k,0] = min{E_n[k,0], E_w};
```

Al termine della computazione $E_n[k,0]$ contiene il costo ottimo dell'albero. Per conoscere la colorazione ottima occorre tenere traccia della colorazione della radice durante il calcolo del minimo di ogni nodo i nelle righe 11-13. Assumiamo che ogni nodo abbia al più c figli. Per calcolare il costo dell'algoritmo, dobbiamo contare quanti modi ci sono di partizionare l'upper bound m in al più c parti con somma m, utilizziamo dunque le combinazione con ripetizione di c figli presi a m a m; le permutazioni sono dunque

$$\binom{m+c-1}{m} = \binom{m+c-1}{c-1} \leq \binom{k+c-1}{c-1}$$

L'algoritmo costruisce $O(n)$ tabelle ognuna con $O(nk)$ entry. Il costo per calcolare ogni entry, dovuto alle righe 11-13, è

$$\begin{aligned} O\left(\binom{k+c-1}{c-1} c\right) &= O\left(\frac{(k+c-1)^{c-1}}{(c-1)!} c\right) = \\ &= O((\max\{k,c\})^{c-1}) \end{aligned} \tag{2.2.5}$$

La riga 15 controlla il caso in cui l'ottimo sia dato da soli nodi bianchi, poiché questo non è incluso nelle righe precedenti. Tale controllo impiega solo tempo $O(n)$. Ciò premesso, la complessità dell'algoritmo è $O(kn^2(\max\{k, c\})^{c-1})$.

2.2.1.2 Correttezza e ottimalità

Per dimostrare la correttezza dell'algoritmo, occorre mostrare che la Definizione 2.2.3 calcoli in modo corretto il valore $E_i(l, m)$, cioè che conteggi i costi energetici in modo consistente con la Definizione 2.2.2. Noi supporremo che tale costo sia già ben definito.

Per dimostrare l'ottimalità occorre mostrare la sottostruttura ottima, cioè palesare che nelle Equazioni 2.2.3 e 2.2.4 partendo da sottocasi ottimi si giunge al valore di $E_i(l, m)$ ottimo.

Dimostrazione. Procediamo assumendo che $E_i(l, m)$ sia ottimo.

Studiamo prima il caso in cui i è un nodo bianco non foglia, in questa occasione useremo l'Equazione 2.2.3; sia fissata la permutazione $P(m)$ che produce il costo minimo, e sia $E_j\left(m_j^p, l+1\right)$ il costo di uno specifico sottoalbero selezionato da tele permutazione per minimizzare $E_i(l, m)$. Supponiamo ora che $E_j\left(m_j^p, l+1\right)$ non sia ottimo: potremmo allora sostituire il suo valore ottimo $E'_j\left(m_j^p, l+1\right)$ all'interno dell'Equazione 2.2.3, con la permutazione $P(m)$ precedentemente fissata, ottenendo così un valore $E_i(l, m)$ inferiore all'ottimo, e ciò è ovviamente assurdo. Ne segue che $E_i(l, m)$ è ottimo (con i nodo bianco) se sono ottimi tutti i sottocasi considerati.

Il caso in cui i sia un nodo nero si dimostra nello stesso identico modo.

□

2.2.2 Albero regolare completo

Supponiamo ora che l'albero di input sia un albero regolare completo, dove tutti i nodi non foglia hanno esattamente c figli, e tutte le foglie si trovano alla stessa profondità. Per ottenere un algoritmo più efficiente modifichiamo l'Algoritmo 11 per sfruttare le regolarità dell'albero. Questo algoritmo è stato proposto da *Bo Sheng et al.* [7].

Ogni sottoalbero di un albero regolare è anch'esso un albero regolare, ed i sottoalberi radicati alla stessa profondità hanno la stessa topologia. Questo suggerisce, che invece di mantenere una tabella per ogni sottoalbero come nell'Algoritmo 11, si può tenere una sola tabella per ogni profondità dell'albero. Nominiamo tutti i livelli dell'albero dalle foglie fino alla radice in questo modo: tutte le foglie sono al livello 0, i loro padri al livello 1, la radice al livello $\lfloor \log_c n \rfloor$, e tutti gli altri di conseguenza. Per ogni livello h, definiamo una tabella bi-dimensionale $E_h[m, l]$ per $0 \leq m \leq k$ e $0 \leq l \leq \lfloor \log_c n \rfloor - 1$, che rappresenta il costo energetico del sottoalbero T_h con radice al livello h sommato al costo per far giungere le informazioni generate in T_h alla radice. Come nell'algoritmo rpecedente, m è il massimo numero di nodi neri in T_h, ed l il numero di nodi bianchi tra la radice del sottoalbero ed il più vicino antenato nero. Se $m > 0$ allora assumiamo che il sottoalbero contenga almeno un nodo nero.

Definiamo prima la variabile che rappresenta il costo di invio dell'interrogazione da parte di un nodo qualsiasi, quando il sottoalbero radicato in tale nodo ha al più m nodi neri:

$$Q_0(m) = \begin{cases} 0 & \text{se } m = 0 \\ \frac{e_{tr} + c e_{re}}{e_{tr} + e_{re}} & \text{se } m \geq 1 \end{cases}, Q_1(m) = Q_0(m-1) \tag{2.2.6}$$

Sia inoltre $H = \lfloor \log_c n \rfloor$. L'algoritmo procede dal basso verso l'alto, calcolando prima $E_h[m, l]$ per le foglie al livello 0, per $0 \leq m \leq k$ e $0 \leq l \leq H - 1$. Poi calcola $E_h[m, l]$ verso l'alto, dal livello 1 al livello $H - 1$. La radice è trattata separatamente poiché sarà sempre un nodo nero.

2.2.2.1 Pseudocodice

Algoritmo 12: Modello 1, limitato, albero regolare completo

```
1  H = ⌊log_c n⌋;
2  for m = 0 to k do
3  |  for l = 0 to H − 1 do
4  |  |  if m = 0 then
5  |  |  |  E_0[m, l] = (l + 1)r_d s_d + (H − l)r_q α s_d;
6  |  |  if m ≥ 1 then
7  |  |  |  E_0[m, l] = (H + 1)r_q α s_d
   |  |  end
   |  end
   end
8  for h = 1 to H − 1 do
9  |  for m = 0 to k do
10 |  |  for l = 0 to H − 1 do
   |  |  |  // i bianco
11 |  |  |  min_1 = min_{∀p∈P(m)} {Σ_{j=1}^{c} E_{h−1}[m_j^p, l + 1]}+
   |  |  |       +(l + 1)r_d s_d + (H − h − l)r_q α s_d + Q_0(m);
   |  |  |  // i nero
12 |  |  |  min_2 = min_{∀p∈(P(m−1))} {Σ_{j=1} E_{h−1}[m_j^p, 0]}+
   |  |  |       +(H − h + 1)r_q α s_d + Q_1(m);
13 |  |  |  E_h[m, l] = min{min_1, min_2};
   |  |  end
   |  end
   end
14 E_H[k, 0] = min_{∀p∈P(k−1)} { Σ_{j=1}^{c} E_{H−1} [m_j^p, 0] } + r_q α s_d + Q_1(m);
```

L'algoritmo costruisce una tabella per ogni livello, cioè $\lfloor \log_c n \rfloor$ tabelle di dimensione $O(k) \times O(\log n)$. Come visto nell'Equazione 2.2.5 il costo per calcolare i minimi tenendo conto del numero di permutazioni $p(m)$ e del costo delle sommatorie al loro interno ha come upperbound $\max\{k, c\}^{c-1}$. I cicli *For* vengono invece eseguiti $O(k(\log^2 n))$ volte. L'algoritmo presentato quindi ha complessità

$$O\left(k(\log^2 n)(\max\{k, c\})^{c-1}\right) \tag{2.2.7}$$

Per individuare la colorazione ottima è necessario tenere traccia del colore di i che minimizza $E_h[m, l]$ nel ciclo più interno, e del colore dei suoi sottoalberi.

Mostriamo ora come sia possibile ottenere un diverso costo computazionale per questo algoritmo. Osserviamo che poiché tutti i figli di un nodo sono tra loro identici non ci interessa come disporre tra loro le diverse quantità in cui viene suddiviso l'upperbound m. Facciamo

un esempio: per $c = 3, m = 3$ le permutazioni generate da $P(m)$ sono:

$$\begin{array}{c}(0,0,3)\\(0,3,0)\\(3,0,0)\\(1,0,2)\\(1,2,0)\\(0,1,2)\\(2,1,0)\\(0,2,1)\\(2,0,1)\\(1,1,1)\end{array}$$

Ma tra queste possiamo considerare ad esempio $(1,0,2) = (1,2,0)$ poiché non ci interessa quale tra i figli avrà 0 o 1 nodo nero, i figli sono cioè indistinti. In altre parole, nel caso di un albero regolare, non importa l'ordine dei valori all'interno della permutazione. Ciò che cerchiamo non è dunque il numero di permutazioni ma il numero di **partizioni** di m in c parti, dato dalla funzione $P(m)$ [13, Sec7.2.1.4]. Dalla stessa fonte conosciamo un limite superiore del numero di partizioni[4]

$$P(m) \leq \frac{\frac{\pi}{\sqrt{6}} e^{\left(\frac{2\pi}{\sqrt{6}}\sqrt{m}\right)}}{\sqrt{m}}$$

Esiste inoltre un algoritmo per generare le partizioni di m in c parti in tempo $O(P(m) + c)$ [13, Sec7.2.1.4 Pag.39]. Utilizzando tale algoritmo per generare in anticipo tutte le partizioni $P(m), \forall m \in [1, \dots k]$ in c parti, avremo un costo iniziale di

$$O\left(k\left[\frac{\frac{\pi}{\sqrt{6}} e^{\left(\frac{2\pi}{\sqrt{6}}\sqrt{k}\right)}}{\sqrt{k}} + c\right]\right)$$

ed un costo per la parte centrale dell'algoritmo pari a

$$O\left(k(\log^2 n)\frac{\frac{\pi}{\sqrt{6}} e^{\left(\frac{2\pi}{\sqrt{6}}\sqrt{k}\right)}}{\sqrt{k}}\right)$$

Per un costo totale di

$$O\left(k(\log^2 n)\frac{\frac{\pi}{\sqrt{6}} e^{\left(\frac{2\pi}{\sqrt{6}}\sqrt{k}\right)}}{\sqrt{k}} + kc\right) \tag{2.2.8}$$

Che può essere paragonato a quello espresso nell'Equazione 2.2.7 per scegliere l'algoritmo migliore. In particolare questo costo sarà migliore quando $c >> k$.

Correttezza ed ottimalità dell'algoritmo derivano banalmente dalle osservazioni sulla struttura dell'albero regolare completo e dalle dimostrazioni di correttezza ed ottimalità dell'Algoritmo 11 trattato nella Sezione 2.2.1.

[4] Questo tiene conto nel numero di partizioni in $[1, 2, \dots, m]$ parti, e non solo in c parti. Se $c \leq m$ allora generiamo anche tutte le partizioni in c parti, se invece $c > m$, le restanti $m - c$ posizioni saranno riempite con degli 0 e non influiranno, poiché, lo ricordiamo, non importa l'ordine nei valori all'interno delle permutazioni: le permutazioni $(1,0,2,0)$ e $(1,2,0,0)$ saranno cioè considerate identiche.

2.2.3 Cammino

Studiamo il caso in cui il grafo in ingresso sia un cammino composta da n nodi. Possiamo innanzitutto conoscere in anticipo quanti nodi neri dovremo disporre:

Teorema 2.2.4. *Nonostante il problema sia di utilizzare al più k nodi, l'ottimo si ottiene con esattamente k nodi.*

Dimostrazione. Compariamo k con il numero di nodi neri k_u utilizzati nella soluzione ottima nel caso unlimited. Se $k \geq k_u$ allora il costo totale aumenterà se useremo più di k_u nodi neri, dunque ne dovremo utilizzare esattamente k_u ed entrambi i problemi avranno la stessa soluzione, cioè la medesima colorazione ottima. Se invece $k < k_u$, assumiamo per assurdo che esattamente $\overline{k}$ nodi neri siano utilizzati nella soluzione ottima S nel caso limitato, e che sia $\overline{k} < k$. In S i nodi $0, 1, \ldots, \overline{k} - 1$ devono essere neri, poiché se qualcuno di essi fosse bianco, il costo di S non sarebbe ottimo (e potremmo sempre aggiungere neri, poiché $\overline{k} < k$). Ma ricordiamo che la soluzione ottima nel caso non limitato k_u nodi neri con $k_u > k > \overline{k}$ e per il Teorema 2.1.7 il cammino gode di proprietà bitonica, dunque esiste una seconda soluzione S' con k nodi neri, e costo compreso tra il costo S e quello dello soluzione nel caso non limitato. In S' tutti i nodi $0, 1, \ldots, k - 1$ sono neri, ed il costo di S' è minore di S, ma ciò è assurdo poiché abbiamo supposto S ottima. □

Per risolvere il problema, assumiamo di conoscere che x sia il nodo nero a profondità massima, e troveremo di conseguenza il costo ottimo con tale assunzione. Tra tutte le possibili x tali che $k - 1 \leq x \leq n - 1$ troveremo il valore x^* che minimizza la funzione di costo.

Per definire la funzione di costo, calcoliamo il costo che incorre in ogni nodo i. I dati grezzi generati da i bianco devono attraversare un certo numero di nodi bianchi fino ad arrivare all'antenato nero più prossimo a_i, con un costo $(i - a_i)r_d s_d$; da qui verso la radice questi dati sono trasferiti come messaggi di risposta ad un costo di $a_i \alpha r_q s_d$. Il costo per ogni nodo i è dunque

$$(i - a_i)r_d s_d + a_i \alpha r_q s_d$$

Notiamo che se i è un nodo nero, allora $a_i = i$. Inoltre a_i è sempre ben definito poiché la radice è un sempre nera. Dobbiamo inoltre considerare il costo di diffusione dell'interrogazione, che deve essere inviata e trasmessa fino al nodo x, dunque avremo un costo di diffusione $x r_q s_q$.

Riassumendo, sapendo che x è l'ultimo nodo nero, il costo energetico totale diventa:

$$\begin{aligned}
C &= \sum_{i=0}^{n-1}[(i-a_i)r_d s_d + a_i \alpha r_q s_d] + x r_q s_q = \\
&= \sum_{i=0}^{x}[(i-a_i)r_d s_d + a_i \alpha r_q s_d] + x r_q s_q + \sum_{j=x+1}^{n-1}[(j-x)r_d s_d + x\alpha r_q s_d] = \\
&= \sum_{i=0}^{x}[(i-a_i)r_d s_d + a_i \alpha r_q s_d] + x r_q s_q + \\
&\quad + \frac{(n-1-x)(n-x)}{2} r_d s_d + x(n-1-x)\alpha r_q s_d = \\
&= \sum_{i=0}^{x}[(i-a_i)(r_d s_d + \alpha r_q s_d - \alpha r_q s_d) + a_i \alpha r_q s_d] + x r_q s_q + \\
&\quad + \frac{(n-1-x)(n-x)}{2} r_d s_d + x(n-1-x)\alpha r_q s_d = \\
&= \sum_{i=0}^{x} i\alpha r_q s_d + \sum_{i=0}^{x}(i-a_i)(r_d s_d - \alpha r_q s_d) + x r_q s_q + \\
&\quad + \frac{(n-1-x)(n-x)}{2} r_d s_d + x(n-1-x)\alpha r_q s_d = \\
&= \sum_{i=0}^{x}(i-a_i)(r_d s_d - \alpha r_q s_d) + \frac{x(x+1)}{2}\alpha r_q s_d + x r_q s_q + \\
&\quad + \frac{(n-1-x)(n-x)}{2} r_d s_d + x(n-1-x)\alpha r_q s_d
\end{aligned} \tag{2.2.9}$$

Dato x dobbiamo minimizzare questo costo scegliendo opportunamente la disposizione dei nodi neri, che influiscono sui valori a_i. Sia ora l'insieme dei nodi neri

$$S = \{i_0, \ldots, i_{k-1}, | i_0 = 0 < i_1 < \ldots < i_{k-1} = x\}$$

con $a_i = i_j$ quando $i_j \leq i < i_{j+1}$, cioè quando i è compreso tra i nodi neri i_j e i_{j+1}. Minimizzare C nell'Equazione 2.2.9 equivale a minimizzare il primo termine C'

$$\begin{aligned}
C' &= \sum_{i=0}^{x}(i-a_i)(r_d s_d - \alpha r_q s_d) = \\
&= \sum_{j=0}^{k-2}\sum_{i=i_j}^{i_{j+1}-1}(i-i_j)(r_d s_d - \alpha r_q s_d) = \\
&= \sum_{j=0}^{k-2}\frac{\sigma_j(\sigma_j-1)}{2}(i-i_j)(r_d s_d - \alpha r_q s_d) = \\
&= \frac{r_d s_d - \alpha r_q s_d}{2}\sum_{j=0}^{k-2}(\sigma_j^2 - x)
\end{aligned} \tag{2.2.10}$$

dove $\sigma_j = i_{j+1} - i_j$ è la dimensione del j-esimo gruppo costituito da un nodo nero seguito da nodi bianchi. Un esempio di soluzione si può trovare in Figura 2.2.2. Si noti che il contributo

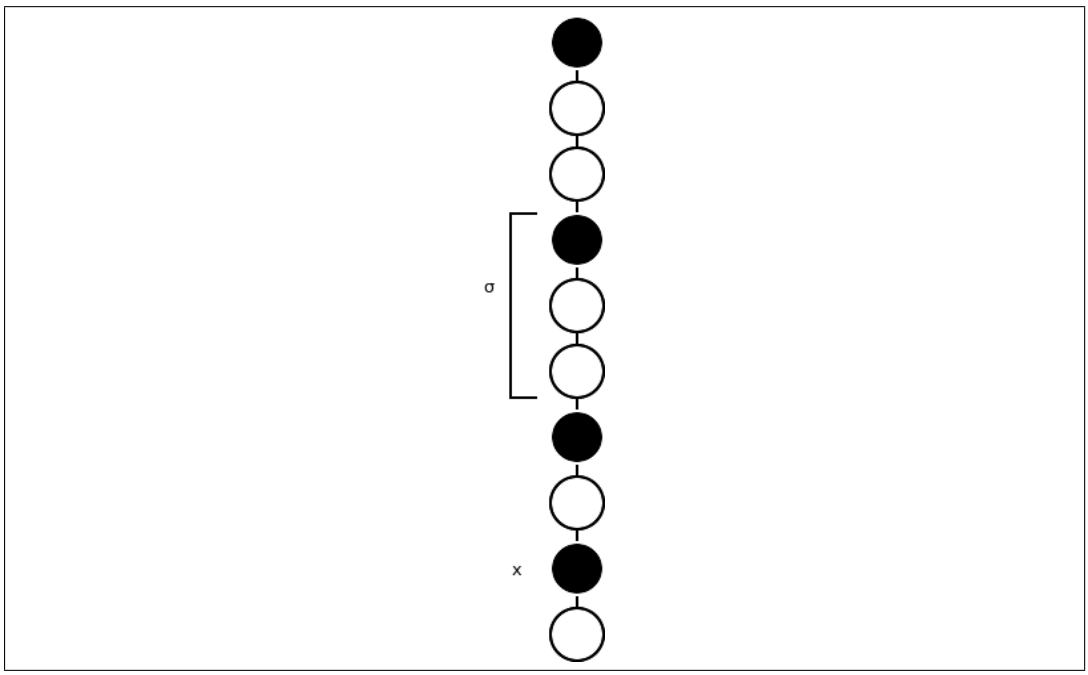


Figura 2.2.2: Esempio di soluzione ottima su cammino nel caso limitato.

del gruppo j dipende sollo dalla dimensione del gruppo σ_j e non dalla posizione relativa nel cammino.

Dopo alcune manipolazioni algebriche è possibile riscrivere l'equazione per C come:

$$\begin{aligned} C =& \frac{r_d s_d - \alpha r_q s_d}{2} \sum_{j=0}^{k-2} (\sigma_j^2) + \left[\frac{-x + (n-x)(n-1-x)}{2} \right] r_d s_d + x r_q s_q + \\ &+ x \left(n - \frac{x}{2} \right) \alpha r_q s_d \end{aligned} \tag{2.2.11}$$

Da qui si vede che minimizzare C equivale a scegliere la disposizione nei nodi neri che minimizza $\sum_{j=0}^{k-2}(\sigma_j^2)$ soggetta a $\sum_{j=0}^{k-2}(\sigma_j^2) = x$ e $\sigma_j \in \mathbb{N}^+$ per $0 \leq j \leq k+2$. Mostreremo ora che la soluzione ottima che minimizza tale costo è:

$$\sigma_j^* = \begin{cases} \lceil \frac{x}{k-1} \rceil & 0 \leq j \leq |x|_{k-1} - 1 \\ \\ \lfloor \frac{x}{k-1} \rfloor & |x|_{k-1} \leq j \leq k-2 \end{cases} \tag{2.2.12}$$

dove $|x|_y$ denota l'operatore modulo [5],[6].

Dimostrazione. Per dimostrare che questo minimizza l'Equazione 2.2.11 assumiamo senza perdita di generalità che i gruppi nella soluzione ottima abbiamo dimensione $b_0, \ldots, b_{k-2}$

[5] Osserviamo che crea gruppi di due sole dimensioni, che differiscono al più di un unità.

[6] Tale problema di minimizzazione è equivalente al problema di dividere un segmento di lunghezza x in $k-1$ parti, volendo minimizzare la somma dei quadrati dei segmenti così creati; per tale problema il minimo si trova tagliando il segmento principale in parti uguali. Allo stesso modo la soluzione proposta nell'Equazione 2.2.12 crea gruppi di dimensione il più possibile uguale tra loro.

di modo che $b_j = \sigma_j^* + t_j$, dove $t_j \in \mathbb{Z}$ e $\sum_{j=0}^{k-2} t_j = 0$. Poiché come abbiamo detto, il costo C dipende dalla dimensione dei gruppi e non dalla loro posizione, assumiamo anche $b_0 \geq b_1 \geq \ldots \geq b_{k-2}$. Da questo e dall'Equazione 2.2.12 deriva anche $t_0 \geq t_1 \geq \ldots \geq t_{|x|_{k-1}-1}$. Osserviamo innanzitutto che

$$\sum_{j=0}^{|x|_{k-1}-1} t_j \geq 0$$

Infatti assumendo per assurdo $\sum_{j=0}^{|x|_{k-1}-1} t_j < 0$, seguirebbe che $b_{|x|_{k-1}-1} = \lceil \frac{x}{k-1} \rceil + t_{|x|_{k-1}-1}$ con $t_{|x|_{k-1}-1} < 0$, poiché per ipotesi $t_0 \geq t_1 \geq \ldots \geq t_{|x|_{k-1}-1}$. Sempre per $t_0 \geq t_1 \geq \ldots \geq t_{|x|_{k-1}-1}$ avendo $t_{|x|_{k-1}-1} < 0$ segue che $\sum_{j=|x|_{k-1}}^{k-2} t_j < 0$. Ma essendo vincolati da $\sum_{j=0}^{k-2} t_j = 0$ è impossibile cha valgano contemporaneamente $\sum_{j=0}^{|x|_{k-1}-1} t_j < 0$ e $\sum_{j=|x|_{k-1}}^{k-2} t_j < 0$.

Per dimostrare che l'Equazione 2.2.12 definisce l'ottimo occorre dimostrare che

$$\sum_{j=0}^{k-2} b_j^2 \geq \sum_{j=0}^{k-2} (\sigma_j^*)^2 \tag{2.2.13}$$

infatti, il nostro problema era minimizzare $\sum_{j=0}^{k-2}(\sigma_j^2)$, ed essendo b_j le dimensione ottime si avrà

$$\sum_{j=0}^{k-2} b_j^2 \leq \sum_{j=0}^{k-2} (\sigma_j^*)^2$$

da cui

$$\sum_{j=0}^{k-2} b_j^2 = \sum_{j=0}^{k-2} (\sigma_j^*)^2$$

Procediamo quindi calcolando questa quantità che ci servirà successivamente.

$$\begin{aligned}
\sum_{j=0}^{k-2} \sigma_j^* t_j = \sum_{j=0}^{|x|_{k-1}-1} \left\lceil \frac{x}{k-1} \right\rceil t_j + \sum_{j=|x|_{k-1}}^{k-2} \left\lfloor \frac{x}{k-1} \right\rfloor t_j = \\
= \left\lfloor \frac{x}{k-1} \right\rfloor \sum_{j=0}^{k-2} t_j + \left(\left\lceil \frac{x}{k-1} \right\rceil - \left\lfloor \frac{x}{k-1} \right\rfloor \right) \sum_{j=0}^{|x|_{k-1}-1} t_j \geq \\
\geq \left(\left\lceil \frac{x}{k-1} \right\rceil - \left\lfloor \frac{x}{k-1} \right\rfloor \right) \sum_{j=0}^{|x|_{k-1}-1} t_j \geq 0
\end{aligned}$$

Possiamo ora dimostrare l'ottimalità tramite la seguente equazione

$$\sum_{j=0}^{k-2} b_j^2 = \sum_{j=0}^{k-2} (\sigma_j^* + t_j)^2 = \sum_{j=0}^{k-2} (\sigma_j^*)^2 + \sum_{j=0}^{k-2} t_j^2 + 2 \sum_{j=0}^{k-2} \sigma_j^+ t_j \geq \sum_{j=0}^{k-2} (\sigma_j^*)^2$$

e questo è ciò che volevamo dimostrare nell'Equazione 2.2.13 □

Possiamo quindi ricavare il costo della soluzione ottima in tempo costante calcolando grazie all'Equazione 2.2.12

$$\sum_{j=0}^{k-2}(\sigma_j^*)^2 = |x|_{k-1}\left\lceil\frac{x}{k-1}\right\rceil^2 + (k-1-|x|_{k-1})\left\lfloor\frac{x}{k-1}\right\rfloor^2 \tag{2.2.14}$$

e sostituendo tale valore all'interno dell'Equazione 2.2.11.

Cerchiamo ora di determinare in tempo sublineare il valore ottimo di x, che ottimizza l'Equazione 2.2.11. sia $C(x)$ la funzione obiettivo quando l'ultimo nodo nero è x e definiamo

$$\triangle C_{x+1} = C(x+1) - C(x)$$

si noti che muovendo l'ultimo nodo nero da x a $x+1$ risulta in un guadagno se $\triangle C_{x+1} < 0$ e in una perdita se $\triangle C_{x+1} > 0$. Spostando verso il basso di una posizione l'ultimo nodo nero, i gruppi definiti nell'Equazione 2.2.12 cambiano leggermente: il gruppo di dimensione più piccola aumenta la sua dimensione di 1, mentre il numero di gruppi rimane invariato. Facciamo presente che in questo cambio di dimensioni, può anche cambiare il numero di gruppi di dimensioni maggiore o minore, ma rimane costante l'effetto di far aumentare di una unità il gruppo più piccolo. In Figura 2.2.3 possiamo vedere come cambiano i gruppi all'aumentare di x.
Da qui si intuisce come il contributo dato a $\triangle C_{x+1}$ dalla somma dei quadrati dell'Equazione 2.2.14 è di

$$\left(2\left\lfloor\frac{x}{k-1}\right\rfloor + 1\right)\frac{r_d s_d - \alpha r_q s_d}{2} > 0 \tag{2.2.15}$$

Se infatti fosse $\left\lfloor\frac{x}{k-1}\right\rfloor = \left\lceil\frac{x}{k-1}\right\rceil$, come nel caso (b) della Figura 2.2.3, allora $\left\lceil\frac{x+1}{k-1}\right\rceil = \left\lceil\frac{x}{k-1}\right\rceil + 1$, dunque il contributo a $\triangle C_{x+1}$ sarebbe di

$$\left\lceil\frac{x+1}{k-1}\right\rceil^2 - \left\lceil\frac{x}{k-1}\right\rceil^2 = \left(\left\lceil\frac{x}{k-1}\right\rceil + 1\right)^2 - \left\lceil\frac{x}{k-1}\right\rceil^2 = 2\left\lfloor\frac{x}{k-1}\right\rfloor + 1$$

Se invece $\left\lceil\frac{x}{k-1}\right\rceil = \left\lfloor\frac{x}{k-1}\right\rfloor + 1$, il contributo a $\triangle C_{x+1}$ sarebbe sempre di

$$\left\lceil\frac{x}{k-1}\right\rceil^2 - \left\lfloor\frac{x}{k-1}\right\rfloor^2 = \left(\left\lfloor\frac{x}{k-1}\right\rfloor + 1\right)^2 - \left\lfloor\frac{x}{k-1}\right\rfloor^2 = 2\left\lfloor\frac{x}{k-1}\right\rfloor + 1$$

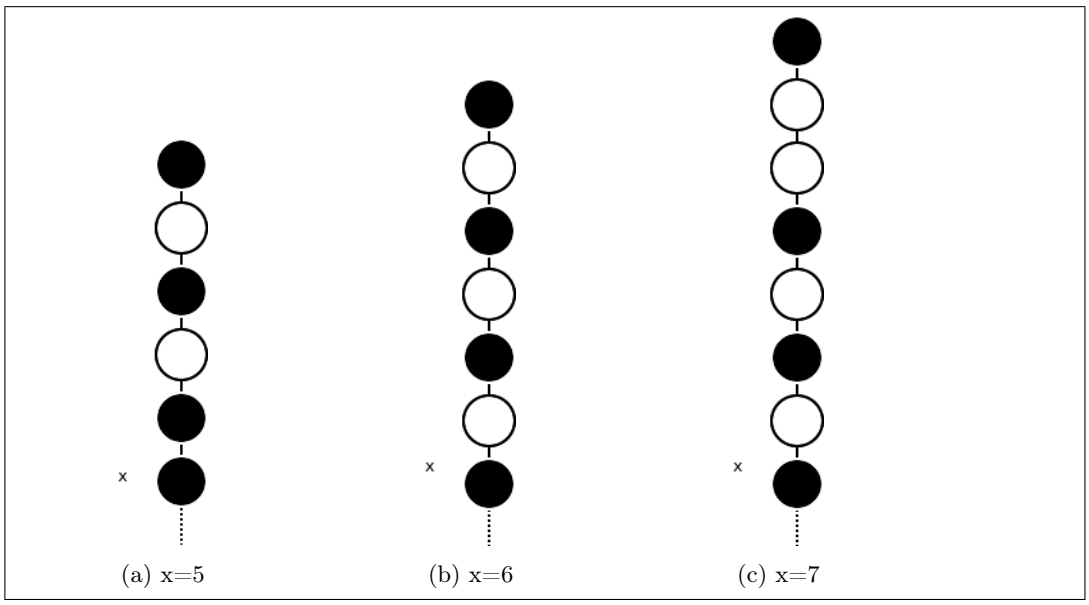
Riscrivendo l'Equazione 2.2.11 per $x+1$ e x si ottiene:

$$\begin{aligned}\triangle C_{x+1} =& \left(2\left\lfloor\frac{x}{k-1}\right\rfloor + 1\right)\frac{r_d s_d - \alpha r_q s_d}{2} + \left(n - x - \frac{1}{2}\right)(\alpha r_q s_d - r_d s_d) + r_q s_q \\ =& (\alpha r_q s_d - r_d s_d)\left(1 + \left\lfloor\frac{x}{k-1}\right\rfloor - n + x\right) + r_q s_q\end{aligned} \tag{2.2.16}$$

che è strettamente crescente in x. Dunque $C(x)$ ha un andamento bitonico: facendo scendere l'ultimo nero x, il costo prima descresce e poi decresce.

Possiamo quindi calcolare il valore ottimo di x guardando in quale momento $\triangle C_{x+1}$ diventa positiva per la prima volta. Sostituendo $m = x(k-1)$ all'interno dell'Equazione 2.2.16 con $m \in \mathbb{N}, 1 \leq m \leq \left\lfloor\frac{n-1}{k-1}\right\rfloor$ si trova che per

$$m = \left\lceil\frac{n-1}{k} - \frac{r_q s_q}{k(r_d s_d - \alpha r_q s_d)}\right\rceil$$



(a) x=5 (b) x=6 (c) x=7

Figura 2.2.3: Dimensione ottima dei gruppi
Con $x = 5$ abbiamo **2** gruppi di dimensione **2** e **1** gruppo di dimensione **1**
Con $x = 6$ abbiamo **0** gruppi di dimensione **2** e **3** gruppi di dimensione **2**
Con $x = 7$ abbiamo **1** gruppo di dimensione **3** e **2** gruppi di dimensione **2**
in entrambi i casi, aumentando x di 1, il gruppo più piccolo aumenta di 1.

$\triangle C_{x+1}$ diventa positivo. Poiché inoltre $\triangle C_{(m-1)(k-1)} < 0$ e $\triangle C_{m(k-1)} \geq 0$, il valore x^* che minimizza C sarà compreso nell'intervallo

$$(m-1)(k-1) \leq x^* \leq m(k-1)$$

Conludiamo quindi dicendo che il valore ottimo x^* può essere individuato tramite una ricerca binaria su sequenza bitonica in tempo $O(\log k)$. Per individuare le posizioni ottime dei nodi neri tuttavia è necessario tempo $O(k)$, utilizzando l'Equazione 2.2.12. L'insieme S dei nodi neri si può costruire come:

$$S = \bigcup_{0 \leq j \leq |x^*|_k} \left\{ j \left\lceil \frac{x^*}{k-1} \right\rceil \right\} \bigcup_{1 \leq j \leq k-1-|x^*|_k} \left\{ |x^*|_{k-1} \left\lceil \frac{x^*}{k-1} \right\rceil + j \left\lfloor \frac{x^*}{k-1} \right\rfloor \right\}$$

2.2.3.1 Pseudocodice

Mostriamo ora per completezza lo pseudocodice dell'algoritmo.

Algoritmo 13: Modello 1, limitato, cammino

$m = \left\lceil \frac{n-1}{k} - \frac{r_q s_q}{k(r_d s_d - \alpha r_q s_d)} \right\rceil$;
$(m-1)(k-1) \leq x^* \leq m(k-1)$;
$\sum_{j=0}^{k-2} (\sigma_j^*)^2 = |x|_{k-1} \left\lceil \frac{x}{k-1} \right\rceil^2 + (k-1-|x|_{k-1}) \left\lfloor \frac{x}{k-1} \right\rfloor^2$;
$E = +\infty$;
$x^* = 0$;
foreach *nodo* x *where* $(m-1)(k-1) \leq x \leq m(k-1)$ *con ricerca binaria* **do**
    Calcola C con l'Equazione 2.2.11;
    if $C < E$ **then**
        $E = C$;
        $x^* = x$;
    end
end
$S = \bigcup_{0 \leq j \leq |x^*|_k} \left\{ j \left\lceil \frac{x^*}{k-1} \right\rceil \right\} \bigcup_{1 \leq j \leq k-1-|x^*|_k} \left\{ |x^*|_{k-1} \left\lceil \frac{x^*}{k-1} \right\rceil + j \left\lfloor \frac{x^*}{k-1} \right\rfloor \right\}$;

Al termine della computazione E contiene il costo minimo ed s l'insieme di nodi neri. L'algoritmo utilizza tempo $O(\log k)$ per trovare il costo minimo, ma poi è necessario tempo $O(k)$ per costruire l'insieme ottimo di nodi neri e piazzarli effettivamente nell'albero.

2.2.4 Caterpillar

Presentiamo ora un algoritmo diverso dai precedenti per trovare il costo ottimo di un grafo caterpillar con al più k nodi neri. Per la descrizione dei grafi caterpillar si veda la Sezione 2.1.4.

Descriviamo il teorema principale che useremo in seguito:

Teorema 2.2.5. *In un caterpillar, se ignoriamo il costo di discesa dell'interrogazione, conviene sempre mettere un nero nella dorsale piuttosto che in foglia.*

Con 'ignorare il costo di discesa dell'interrogazione' intendiamo che l'interrogazione viene già ricevuta ed inoltrata da tutti i nodi che stiamo considerando. In particolare, essendo nel primo modello e su caterpillar, questo è vero se stiamo trattando i nodi con profondità al più uguale a quella nel nodo nero con profondità massima.

Dimostrazione. La dimostrazione di svolge in due parti, nella prima mostriamo che porre una foglia bianca ed il padre nero è meglio che lasciare la foglia nera ed il padre bianco, nella seconda mostriamo che è meglio posizionare un nodo nero nella dorsale, in un nodo senza figli, piuttosto che in una foglia con padre nero. Si noti che non stiamo in alcun modo ponendo vincoli a come disporre i nodi neri nella dorsale, ma solo che è preferibile far divenire prima la dorsale interamente nera e poi annerire le foglie.

Si osservino i casi in Figura 2.2.4. Il costo del primo caso è sicuramente uguale o migliore al costo del secondo caso. Valutiamo infatti il costo dei singoli nodi con la Definizione 2.2.1, ovvero

$$e(i) = \begin{cases} d_i r_q s_d \alpha & \text{se } i \text{ nero} \\ (l+1) r_d s_d + (d_i - l - 1) r_q s_d \alpha & \text{se } i \text{ bianco} \end{cases}$$

ricordiamo che con questa definizione è possibile risalire al costo totale dell'albero tramite $E = \sum_i e(i) + Q$ dove Q rappresenta i costi per la diffusione dell'interrogazione.

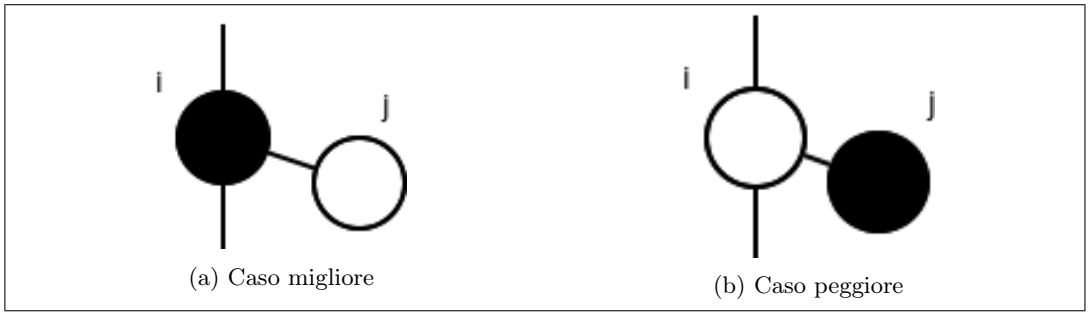
Nella Figura 2.2.4a abbiamo $e(j) = r_d s_d + (d_i) r_q s_q \alpha$ e $e(i) = (d_i) r_q s_q \alpha$, sommandoli otteniamo

$$E' = r_d s_d + (2d_i) r_q s_q \alpha$$

Nella Figura 2.2.4b abbiamo invece $e(j) = (d_i + 1) r_q s_q \alpha$ e $e(i) = (l+1) r_d s_d + (d_i - l - 1) r_q s_d \alpha$ che sommati ci danno

$$E'' = (l+1) r_d s_d + (2d_i - l) r_q s_d \alpha$$

Si nota quindi che $E'' \geq E'$.



(a) Caso migliore (b) Caso peggiore

Figura 2.2.4: Prima parte della dimostrazione

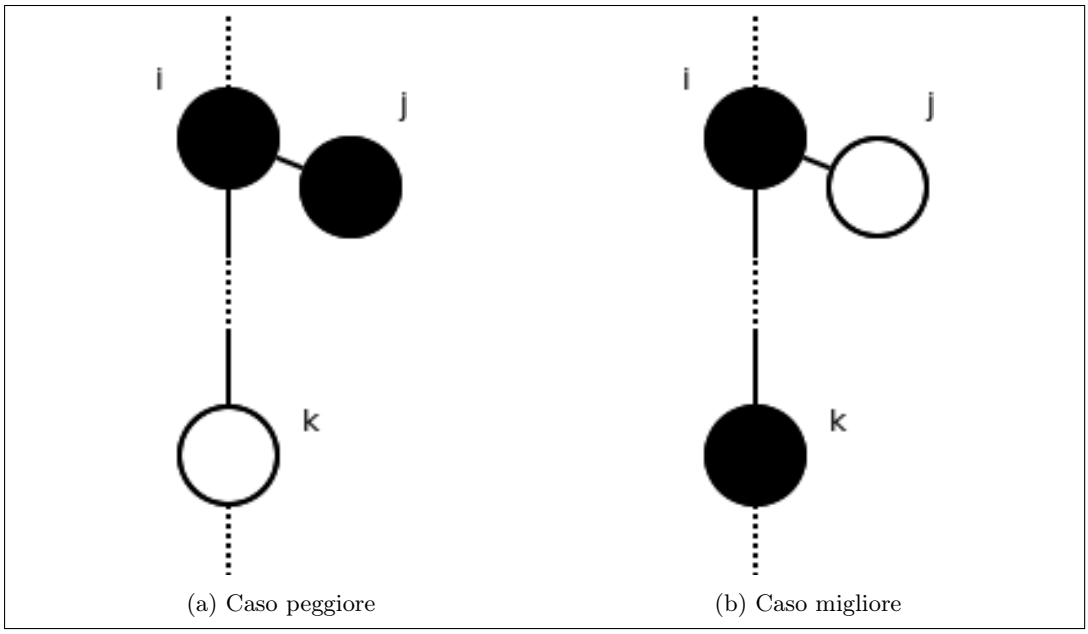
Per la seconda parte, si osservino i due casi in Figura 2.2.5. Il caso con un nodo nero in foglia è sempre peggiore o uguale dell'altro caso. Infatti nella Figura 2.2.5a $e(j) = (d_j)r_q s_q \alpha$ e $e(k) = (l+1)r_d s_d + (d_k - l - 1)r_q s_d \alpha$ che sommati danno

$$E' = (l+1)r_d s_d + (d_j + d_k - l - 1)r_q s_d \alpha$$

Nella Figura 2.2.5b invece $e(j) = (1)r_d s_d + (d_j - 1)r_q s_d \alpha$ e $e(k) = d_k r_q s_d \alpha$ che sommati danno

$$E'' = (1)r_d s_d + (d_j + d_k - 1)r_q s_d \alpha$$

da cui si vede $E'' \leq E'$, ed il ragionamento non cambia se $d_k < d_j$.



(a) Caso peggiore (b) Caso migliore

Figura 2.2.5: Seconda parte della dimostrazione

Notiamo anche che in entrambi i casi, posizionando il nero nella dorsale, diminuirà anche il valore l dei suoi discendenti bianchi, facendo così diminuire ulteriormente il costo totale dell'albero.

□

Definiamo le variabili principali:

Definizione 2.2.6. $E(q, m)$ = costo ottimo del caterpillar $Cat(0, q)$ con al più m neri; non comprende i costi di invio e ricezione dell'interrogazione.

Supponiamo che l'ultimo nodo nero nella dorsale x sia sempre $x > q$ dunque rientriamo nelle ipotesi del Teorema 2.2.5. $E(q, m), \forall q \in [0, h], \forall m \in [2, k]$ è definito dalla seguente relazione di programmazione dinamica:

$$E(q,m) = \begin{cases} B(q,m) & \text{se } m \geq q+1 \\ \min_{\forall x, m \leq x \leq q} \{E(x-1, m-1) + R(x,q)\} & \text{se } m < q+1 \end{cases} \tag{2.2.17}$$

inoltre $E(q, 1) = R(0, q)$.
Nel primo caso avremo $m > q + 1$ e quindi avremo tutta la dorsale nera (per il Teorema

2.2.5) più alcune eventuali foglie nere (ci interessa solo il loro numero, non quali siano nere esattamente); il costo in questo caso è $B(q, m)$ e si può calcolare in tempo costante tramite l'Equazione 2.2.20. Nel secondo caso avremo solo foglie bianche in $Cat(0, q)$ ed alcuni nodi neri nella dorsale per il Teorema 2.2.5. Qui avremo almeno un nodo bianco nella dorsale e tutte le foglie bianche; tra i nodi neri cerchiamo l'ultimo, cioè quel nodo nero x più vicino a q, con $m \leq x \leq q$ e ci riconduciamo al caso definito ricorsivamente. Fissato x possiamo calcolare $E(q, m)$ sommando il costo di $Cat(x, q)$ che avrò solo x nero, e quindi costo $R(x, q)$, e sommando il costo ottimo di $Cat(0, x-1)$ con al più $m-1$ neri nella dorsale, costo questo già conosciuto $E(x-1, m-1)$.

Vediamo come calcolare poi il costo energetico totale E. Sia x il nodo nero a maggiore profondità in tutto l'albero $Cat(0, h)$, esso sarà sempre nella dorsale: se così non fosse potremmo spostarlo nel fratello nella dorsale senza aumentare il costo totale dell'albero. L'algoritmo utilizza la programmazione dinamica per calcolare prima tutti i valori $E(q, m), \forall q \in [0, h], \forall m \in [1, k]$, e poi trova il costo E cercando esaustivamente l'ultimo nodo nero x. Poiché nel calcolo di E si conosce l'ultimo nero x allora è conosciuto anche il costo di discesa dell'interrogazione. Possiamo esprimere il costo totale E come segue:

$$E = \min_{\forall x, x \leq h} \{G(x) + E(x-1, k-1) + R(x, h)\} \tag{2.2.18}$$

dove:

$G(x) =$ costo di discesa dell'interrogazione quando l'ultimo nodo nero è x.

$R(i, j) =$ costo di risalita fino alla radice delle informazioni generate dal sottoalbero $Cat(i, j)$ con i nero e tutti gli altri nodi bianchi.

2.2.4.1 Pseudocodice

Mostriamo ora lo pseudocodice dell'algoritmo:

Algoritmo 14: Modello 1, limitato, caterpillar

```
1  Precomputazioni;
2  for q=0 to h do
3  |  E(q, 1) = R(0, q);
4  end
5  for m=2 to k-1 do
6  |  for q=0 to h do
7  |  |  if m > q + 1 then
8  |  |  |  E(q, m) = B(q, m);
9  |  |  else
10 |  |  |  E(q, m) = min_{∀x, m≤x≤q} {E(x − 1, m − 1) + R(x, q)};
11 |  |  end
12 |  end
13 end
14 E = min_{∀x, x≤h} {G(x) + E(x − 1, k − 1) + R(x, h)};
```

L'algoritmo utilizza tempo totale $O(h^2 k)$, necessario per effettuare i due cicli innestati e la ricerca del minimo. Durante il calcolo di E ci si imbatte in $x = 0$; in tal caso definendo $E(q, m) = 0 \forall q < 0, m < 0$ viene calcolato il costo corretto $E = R(x, q)$. Per ricostruire la colorazione ottima è necessario tenere traccia delle colorazioni per ogni $E(q, m)$. Spieghiamo come procedere per casi. Sia β la colorazione ottima di $E(q, m)$.

Caso 1. quando $E(q,m) = B(q,m)$, β avrà tutta la dorsale nera, ed altre $m-q-1$ foglie qualsiasi nere. Le altre foglie rimangono bianche.

Caso 2. quando $E(q,m) = \min_{\forall x, m \leq x \leq q}\{E(x-1,m-1)+R(x,q)\}$, β avrà il nodo x nero, ed i suoi discendenti fino a q saranno bianchi. Gli altri nodi avranno la stessa colorazione di $E(x-1,m-1)$.

La colorazione ottima di tutto l'albero si ottiene durante il calcolo di $E = \min_{\forall x, x \leq h}\{r_q s_q G(x) + E(x-1,k-1) + R(x,h)\}$. In questo caso x sarà il nodo nero a profondità maggiore, ed i restanti nodi avranno la stessa colorazione di $E(x-1,k-1)$.

2.2.4.2 Dimostrazione di ottimalità e correttezza

La dimostrazione prova che $E(q,m)$ è ben definito ed ottimo. Occorrerebbe anche mostrare che sia corretta ed ottima l'Equazione 2.2.18 di E, e le definizioni di $B(q,m)$, $G(x)$ e $R(i,j)$, ma qui le assumeremo già corrette. Riscriviamo per comodità la definizione

$$E(q,m) = \begin{cases} B(q,m) & \text{se } m > q+1 \\ \min_{\forall x, x \leq q}\{E(x-1,m-1)+R(x,q)\} & \text{Altrimenti} \end{cases}$$

ed ancora $E(q,1) = R(0,q)$.

Dimostrazione. Procediamo per induzione su m. Per $m = 1$ possiamo avere solo la radice nera, dunque il costo di $E(q,1)$ coincide con la definizione di $R(0,q)$; questo caso è dunque corretto assumendo che sia ben definita $R(0,q)$. Per $m > 1$ abbiamo, per ipotesi induttiva, che $E(q,m-1)$ è ben definito ed ottimo per qualsiasi q. si distinguano altri due casi:

Caso 1. $m \geq q+1$, dunque per il Teorema 2.2.5 tutta la dorsale sarà nera ed il suo costo si calcola con $B(q,m)$. Dunque questo caso è corretto se è ben definito $B(q,m)$.

Caso 2. $m < q+1$, per il Teorema 2.2.5 non avremo foglie nere. Al suo interno cerchiamo esaustivamente l'ultimo nero x. Una volta individuato, il costo è dato da un caterpillar, che si estende da x alle foglie di q in cui solo x è nero dunque il suo costo si calcola con $R(x,q)$, e da un secondo caterpillar $Cat(0,x-1)$ il cui costo è $E(x-1,m-1)$ che conosciamo già per ipotesi induttiva. Possiamo anche mostrare la sottostruttura ottima, cioè che si giunge sempre ad $E(q.m)$ ottimo se i sottocasi sono ottimi. Infatti fissato x se $E(x-1,m-1)$ non fosse ottimo potremmo sostituire il suo valore ottimo $E'(x-1,m-1) < E(x-1,m-1)$ nel calcolo di $E'(q,m) = E'(x-1,m-1) + R(x,q)$ ottenendo un nuovo costo minore di quello iniziale $E'(q,m) < E(q,m)$ giungendo così all'assurdo.

□

2.2.4.3 Precomputazioni

Ci rimane da mostrare quali precomputazioni eseguire e calcolare $B(q,m)$, $G(x)$ ed $R(i,j)$ in tempo costante all'interno dell'Algoritmo 14. Riprendiamo le definizioni precedenti e quelle definite nella Sezione 2.1.4 poiché serviranno durante la trattazione:

$G(x) =$ costo di discesa dell'interrogazione quando l'ultimo nodo nero è x.

$R(i,j) =$ costo delle informazioni generate dal sottoalbero $Cat(i,j)$ dove i è nero e tutti gli altri nodi sono bianchi; il costo comprende la sola risalita delle informazioni fino alla radice.

$B(q,m) =$ costo di $Cat(0,q)$ con al più m neri e $m \geq q+1$.

$N_i =$ numero di nodi in $Cat(i,h)$.

$N'_i =$ numero di nodi in $Cat(0,i)$.

$M_i =$ numero di messaggi scambiati in $Cat(i,h)$, ovvero la risalita di informazioni dalle foglie sino ad i.

$M'_i =$ Numero di messaggi scambiati in $Cat(0,i)$, ovvero la risalita delle informazioni generate dai nodi in $Cat(0,i)$ fino a 0.

I valori $N_i, N'_i, M_i, M'_i, \forall i \in [0,h]$ devono essere calcolati, rispettivamente, mediante gli Algoritmi 5,6,7 e 8 di complessità $O(h)$; tali algoritmi rappresentano tutte e sole le precomputazioni da effettuare nel nostro caso.

È ora possibile calcolare $R(i,j)$ considerando il numero di messaggi M_i scambiati in $Cat(i,h)$, sottraendo il numero di messaggi M_{j+1} scambiati in $Cat(j+1,h)$, sottraendo la risalita fino ad i dei messaggi generati da $Cat(j+1,h)$, e sommando la risalita dei messaggi generati dai $(N_i - N_{j+1})$ nodi di $Cat(i,j)$; ricomponendo gli addendi moltiplicati per gli opportuni coefficienti si ottiene:

$$R(i,j) = [M_i - M_{j+1} - N_{j+1}(j-i+1)]r_d s_d + r_q s_d \alpha (N_i - N_{j+1}) \tag{2.2.19}$$

dunque è possibile calcolare $R(i,j)$ in tempo $O(1)$. La Figura 2.2.6 riporta un esempio per convincersi che l'Equazione 2.2.19 sia corretta.

Calcoliamo ora $B(q,m)$ definito come il costo di $Cat(0,q)$ con al più m neri e $m \geq q+1$. $Cat(0,q)$ contiene esattamente N'_q nodi. Se $m \geq N'_q$ avrò solo nodi neri, altrimenti avrò $N'_q - m$ foglie bianche e tutti gli altri nodi saranno neri per il Teorema 2.2.5. Nel caso $m < N'_q$, fra i M'_q messaggi scambiati, esattamente $N'_q - m$ partono da foglie bianche e giungono al loro padre, mentre tutti gli altri messaggi partono da nodi neri. Possiamo dunque calcolare $B(q,m)$ in tempo costante come segue:

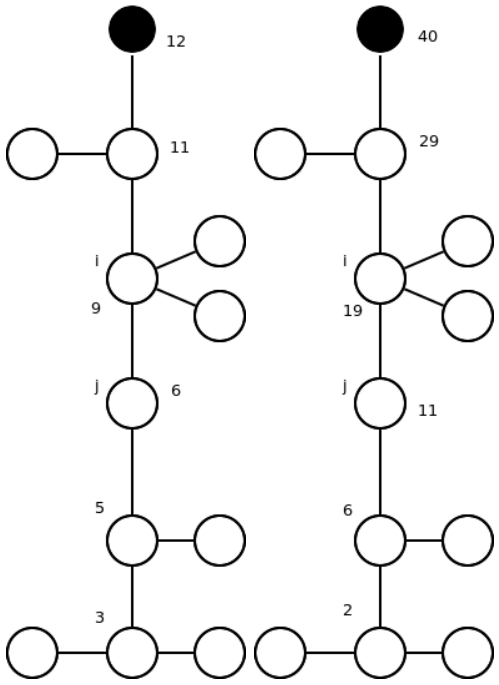
$$B(q,m) = \begin{cases} M'_q r_q s_d \alpha & \text{se } m \geq N'_q \\ (N'_q - m) r_d s_d + [M'_q - (N'_q - m)] r_q s_d \alpha & \text{se } m < N'_q \end{cases} \tag{2.2.20}$$

Ci rimane da mostrare come sia possibile ottenere $G(x)$ (costo di discesa dell'interrogazione per giungere ad x) in tempo costante con la seguente definizione:

$$G(x) = \frac{x e_{tr} + e_{re}(N'_{x-1} + 1)}{e_{tr} + e_{re}} r_q s_q \tag{2.2.21}$$

che corrisponde banalmente all'invio dell'interrogazione da parte di tutti i nodi $i, i \in [0, x-1]$, ed alla ricezione dell'interrogazione di tutti i loro figli.

Come volevamo dimostrare, è dunque possibile calcolare $B(q,m)$, $G(x)$ ed $R(i,j)$ in tempo costante con delle precomputazioni di complessità $O(h)$ che non aumentano la complessità dell'Algoritmo 14.



(a) Valori di N_i (b) Valori di M_i

Nodo i	N_i	M_i
0	12	40
1	11	29
2	9	19
3	6	11
4	5	6
5	3	2

(c) Valori per ogni nodo

Figura 2.2.6: Esempio di calcolo di $R(i,j)$; per $i=2, j=3$, applicando l'Equazione 2.2.19, abbiamo $R(2,3) = [19-6-5(3-2+1)]r_d s_d + 2r_q s_d \alpha(9-5) = 3r_d s_d + 8r_q s_d \alpha$, che è corretto, come si può facilmente verificare.

2.2.5 Caterpillar regolare

Supponiamo ora che l'albero in input sia un caterpillar regolare. Con regolare intendiamo che ogni nodo della dorsale, ha esattamente $c+1$ figli di cui $\boldsymbol{c}$ sono foglie; fa eccezione il nodo h che avrà solo c figli tutti foglia. Si noti che questa è una regolarità diversa da quella espressa nella Sezione 2.1.6 (dove ogni nodo ha invece il medesimo numero di figli), ma con le opportune modifiche entrambi gli algoritmi possono adattarsi ad entrambi i tipi di regolarità. L'algoritmo che presentiamo ora è un adattamento di quello presentato per il cammino nella Sezione 2.2.3; dunque supponiamo di sapere che x è l'ultimo nodo nero nella dorsale. Si noti che il costo di diffusione dell'interrogazione dipende solo dalla profondità di x e non dalla colorazione degli altri nodi, quindi tra gli antenati di x il Teorema 2.2.5 è applicabile. L'algoritmo piazza nel miglior modo possibile i nodi neri nella dorsale nelle posizioni sopra ad x; quando la dorsale è piena, se ha ancora neri a disposizione, li pone nelle foglie iniziando da quelle a profondità minore. Le foglie con padri neri infatti avranno tutte il medesimo costo secondo la Definizione 2.2.1, dunque sono tra loro indistiguibili.

Studieremo prima il caso in cui $x+1 \geq k$ dove tutti i nodi neri saranno nella dorsale per il Teorema 2.2.5. Nel caso $x+1 < k$ invece tutta la dorsale sarà nera fino ad x e ci saranno alcune foglie nere.

2.2.5.1 Prima parte

Quando $x+1 \geq k$ tutti i nodi neri sono nella dorsale, ed avremo solo foglie bianche. Eventualmente tutta la dorsale fino ad x può essere nera. Sia a_i l'antenato nero più prossimo ad i, nel caso i sia nero allora $a_i = i$. Definiamo quindi

Definizione 2.2.7. $C(i) =$ il costo del nodo i e delle sue foglie, supponendo che tutte le sue foglie siano bianche.

Questo è calcolabile sommando:

- $cr_d s_d$: costo di invio dei messaggi grezzi dalle foglie fino ad i,
- $(i-a_i)r_d s_d(c+1)$: la risalita dei $c+1$ messaggi grezzi da i ad a_i,
- $a_i(c+1)r_q s_d \alpha$: la risalita di $c+1$ messaggi di risposta da a_i a 0.

Dunque riassumendo

$$C(i) = (i-a_i)r_d s_d(c+1) + cr_d s_d + a_i(c+1)r_q s_d \alpha \tag{2.2.22}$$

Il costo energetico $E(x)$ dell'albero con ultimo nero x, si ricava sommando:

- $C(i), \forall i \in [0,x]$,
- $\sum_{i=x+1}^{h} [cr_d s_d + (c+1)(i-x)r_d s_d + (c+1)(x)r_q s_d \alpha]$: il costo di risalita fino alla radice delle informazioni generate in $Cat(x+1,h)$,
- $(x)r_q s_q \frac{e_{tr}+(c+1)e_{re}}{e_{tr}+e_{re}}$: il costo di diffusione dell'interrogazione.

Abbiamo dunque:

$$
\begin{aligned}
E(x) =& \sum_{i=0}^{x} C(i) + \sum_{i=x+1}^{h} [cr_d s_d + (c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha] \\
&+ (x) r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}} \\
=& \sum_{i=0}^{x} [(i-a_i) r_d s_d (c+1) + c r_d s_d + a_i (c+1) r_q s_d \alpha] \\
&+ \sum_{i=x+1}^{h} [cr_d s_d + (c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha] \\
&+ (x) r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}} \\
=& \sum_{i=0}^{h} cr_d s_d + (c+1) \sum_{i=0}^{x} [(i-a_i) r_d s_d + a_i r_q s_d \alpha] \\
&+ \sum_{i=x+1}^{h} [(c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha] \\
&+ (x) r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}}
\end{aligned}
\tag{2.2.23}
$$

Conoscendo $E(x)$ per ogni nodo x l'algoritmo banale potrebbe iterare tra tutti gli x cercando il costo ottimo in questo modo:

$$E = \min_{x \in [k-1, h]} \{E(x)\}$$

Cercheremo di ottimizzare l'algoritmo analizzando le proprietà di $E(x)$. Notiamo che per ottimizzare $E(x)$ disponendo in modo opportuno i nodi neri, quindi variando i valori a_i, è sufficiente ottimizzare il termine:

$$
\begin{aligned}
E' =& \sum_{i=0}^{x} [(i-a_i)(r_d s_d) + a_i r_q s_d \alpha] = \\
=& \sum_{i=0}^{x} [(i-a_i)(r_q s_d \alpha + r_d s_d - r_q s_d \alpha) + a_i r_q s_d \alpha] = \\
=& \sum_{i=0}^{x} [i r_q s_d \alpha - a_i r_q s_d \alpha + (i-a_i)(r_d s_d - r_q s_d \alpha) + a_i r_q s_d \alpha] = \\
=& \sum_{i=0}^{x} (i r_q s_d \alpha) + \sum_{i=0}^{x} (i-a_i)(r_d s_d - r_q s_d \alpha)
\end{aligned}
\tag{2.2.24}
$$

che equivale ad ottimizzare il secondo termine. Osserviamo che essendo tutti i neri in dorsale, la colorazione dei nodi $[0, \ldots, x-1]$ è composta da gruppi fatti da un nodo nero seguito da nodi bianchi; riprendiamo quindi quanto scritto nell'Equazione 2.2.10, dove definendo $\sigma_j = i_{j+1} - i_j$ come la dimensione del j-esimo gruppo otteniamo

$$C' = \sum_{i=0}^{x}(i - a_i)(r_d s_d - \alpha r_q s_d) =$$
$$= \frac{r_d s_d - \alpha r_q s_d}{2} \sum_{j=0}^{k-2}(\sigma_j^2 - x)$$

che ci permette di riscrivere per intero la formula per $E(x)$.

$$\begin{aligned}
E(x) = & \sum_{i=0}^{h} c r_d s_d \\
& + (c+1)\left[\sum_{i=0}^{x}(i r_q s_d \alpha) + \frac{r_d s_d - \alpha r_q s_d}{2}\sum_{j=0}^{k-2}(\sigma_j^2 - x)\right] \\
& + \sum_{i=x+1}^{h}[(c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha] \\
& + (x) r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}} \\
= & \sum_{i=0}^{h} c r_d s_d \\
& + (c+1)\left[\sum_{i=0}^{x}(i r_q s_d \alpha) + \frac{r_d s_d - \alpha r_q s_d}{2}\sum_{j=0}^{k-2}(\sigma_j^2 - x)\right] \\
& + (c+1)\sum_{i=x+1}^{h}[(i-x) r_d s_d + (x) r_q s_d \alpha] \\
& + (x) r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}}
\end{aligned}$$

svolgendo dei conti risulta poi:

$$\begin{aligned}
E(x) =&(h+1)cr_d s_d + (c+1)\frac{x(x+1)}{2} r_q s_d \alpha \\
&+ (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \sum_{j=0}^{k-2} (\sigma_j^2 - x) \\
&+ (c+1) \sum_{i=1}^{h-x} [(i) r_d s_d + (x) r_q s_d \alpha] \\
&+ (x) r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}} \\
=&(h+1)cr_d s_d + (c+1)\frac{x(x+1)}{2} r_q s_d \alpha \\
&+ (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \sum_{j=0}^{k-2} (\sigma_j^2) \\
&- x(k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&+ (c+1)\frac{(h-x)(h-x+1)}{2} r_d s_d + (c+1)(h-x)(x) r_q s_d \alpha \\
&+ (x) r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}} \\
=&(h+1)cr_d s_d + (c+1)\frac{x(x+1)}{2} r_q s_d \alpha \\
&+ (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \sum_{j=0}^{k-2} (\sigma_j^2) \\
&- x(k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&+ (c+1)\frac{x(-2h-1) + x^2 + h^2 + h}{2} r_d s_d \\
&+ (c+1)(-x^2 + hx) r_q s_d \alpha \\
&+ (x) r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}}
\end{aligned} \tag{2.2.25}$$

Occorre dunque ottimizzare solo $\sum_{j=0}^{k-2} \sigma_j^2$, il cui minimo è dato dall'Equazione 2.2.12, poiché la sua dimostrazione è applicabile anche a questo caso. Quindi si ha il minimo con

$$\sigma_j^* = \begin{cases} \lceil \frac{x}{k-1} \rceil & 0 \le j \le |x|_{k-1} - 1 \\ \\ \lfloor \frac{x}{k-1} \rfloor & |x|_{k-1} \le j \le k-2 \end{cases} \tag{2.2.26}$$

da cui

$$\sum_{j=0}^{k-2} (\sigma_j^*)^2 = |x|_{k-1} \left\lceil \frac{x}{k-1} \right\rceil^2 + (k - 1 - |x|_{k-1}) \left\lfloor \frac{x}{k-1} \right\rfloor^2 \tag{2.2.27}$$

Da questa definizione possiamo notare che i primi gruppi hanno tutti la stessa dimensione $\lceil \frac{x}{k-1} \rceil$, ed i gruppi che li seguono, hanno tutti la dimensione $\lfloor \frac{x}{k-1} \rfloor$. Ciò è possibile,

poiché dovendo ottimizzare $\sum_{j=0}^{k-2} \sigma_j^2$, non importa la posizione dei gruppi, ma solo la loro dimensione, ed abbiamo quindi scelto di disporli in questo modo.

Riassumendo, nel caso $x + 1 \geq k$, il costo dell'albero $E(x)$ si può trovare in tempo $O(1)$ sostituendo l'Equazione 2.2.27 nell'Equazione 2.2.25 .

Sempre in questo caso, cerchiamo di individuare quale sia x ottimo in tempo sublineare; sia dunque

$$\triangle E_{x+1} = E(x+1) - E(x)$$

si noti che spostando l'ultimo nero da x a $x + 1$ il costo dell'albero aumenta se $\triangle E_{x+1} > 0$ e diminuisce se $\triangle E_{x+1} < 0$, inoltre aumentando x di un unità, uno dei gruppi più piccoli definiti nell'Equazione 2.2.26 aumenta la sua dimensione di 1. Il contributo dato a $\triangle E_{x+1}$ dalla somma dei quadrati è dunque

$$\left(2\left\lfloor \frac{x}{k-1} \right\rfloor + 1\right) \frac{r_d s_d - \alpha r_q s_d}{2}(c+1) > 0$$

come già mostrato nell'Equazione 2.2.15 a pagina 45.

Possiamo quindi riscrivere $\triangle E_{x+1}$ come

$$
\begin{aligned}
&=\triangle E_{x+1} = E(x+1) - E(x) = \\
&=(h+1)cr_d s_d - (h+1)cr_d s_d \\
&\quad + (c+1)\frac{(x+1)(x+2)}{2} r_q s_d \alpha - (c+1)\frac{x(x+1)}{2} r_q s_d \alpha \\
&\quad + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \sum_{j=0}^{k-2} (\sigma_j'^2 - \sigma_j^2) \\
&\quad - (x+1)(k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} + x(k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&\quad + (c+1)\frac{(x+1)(-2h-1) + (x+1)^2 + h^2 + h}{2} r_d s_d \\
&\qquad - (c+1)\frac{x(-2h-1) + x^2 + h^2 + h}{2} r_d s_d \\
&\quad + (c+1)(-(x+1)^2 + h(x+1)) r_q s_d \alpha - (c+1)(-x^2 + hx) r_q s_d \alpha \\
&\quad + (x+1) r_q s_q \frac{e_{tr} + (c+1)e_{re}}{e_{tr} + e_{re}} - (x) r_q s_q \frac{e_{tr} + (c+1)e_{re}}{e_{tr} + e_{re}} = \\
&=(c+1)\frac{x+1}{2} r_q s_d \alpha (x + 2 - x) \\
&\quad + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \left(2 \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) \\
&\quad - (k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&\quad + (c+1)\frac{(-2h-1) + 2x + 1}{2} r_d s_d \\
&\quad + (c+1)(-2x - 1 + h) r_q s_d \alpha \\
&\quad + r_q s_q \frac{e_{tr} + (c+1)e_{re}}{e_{tr} + e_{re}} \\
&=(c+1)(x+1) r_q s_d \alpha + (c+1) x r_d s_d + (c+1)(-2x) r_q s_d \alpha \\
&\quad + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \left(2 \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) \\
&\quad - (k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&\quad + (c+1)\frac{(-2h-1) + 1}{2} r_d s_d \\
&\quad + (c+1)(-1 + h) r_q s_d \alpha \\
&\quad + r_q s_q \frac{e_{tr} + (c+1)e_{re}}{e_{tr} + e_{re}} \\
&=x(c+1)(r_d s_d - r_q s_d \alpha) + (c+1) r_q s_d \alpha \\
&\quad + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \left(2 \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) \\
&\quad - (k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&\quad + (c+1)\frac{(-2h-1) + 1}{2} r_d s_d \\
&\quad + (c+1)(-1 + h) r_q s_d \alpha \\
&\quad + r_q s_q \frac{e_{tr} + (c+1)e_{re}}{e_{tr} + e_{re}}
\end{aligned}
\tag{2.2.28}
$$

da qui si vede che $\triangle E_{x+1}$ è strettamente crescente in x, dunque $E(x)$ ha un comportamento bitonico prima decrescente e poi crescente ed avrà il minimo in corrispondenza del valore x^* che trasforma $\triangle E_{x+1}$ da negativa a positiva. Si può dunque trovare $x^* \in [k-1, h]$ che mininizza $E(x)$ in tempo $O(\log h)$.
Osserviamo inoltre come sia possibile sostituire $x = m(k-1)$ con m intero $1 \leq m \leq \left\lfloor \frac{n-1}{k-1} \right\rfloor$, ottenendo

$$\begin{aligned}
& m(k-1)(c+1)(r_d s_d - r_q s_d \alpha) + (c+1) r_q s_d \alpha \\
& + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2}\left(2\left\lfloor \frac{m(k-1)}{k-1} \right\rfloor + 1\right) \\
& - (k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
& + (c+1)\frac{(-2h-1)+1}{2} r_d s_d \\
& + (c+1)(-1+h) r_q s_d \alpha \\
& + r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}} \\
=& m\left[(k-1)(c+1)(r_d s_d - r_q s_d \alpha) + (c+1)(r_d s_d - r_q s_d \alpha)\right] \\
& + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} + (c+1) r_q s_d \alpha \\
& - (k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
& - (c+1) h r_d s_d \\
& + (c+1)(-1+h) r_q s_d \alpha \\
& + r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}} \\
=& m\left[(k-1)(c+1)(r_d s_d - r_q s_d \alpha) + (c+1)(r_d s_d - r_q s_d \alpha)\right] \\
& + (2-k)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
& - (c+1) h r_d s_d \\
& + (c+1) h r_q s_d \alpha \\
& + r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}}
\end{aligned} \tag{2.2.29}$$

Ponendo a zero questa equazione si trova che:

$$m = \left\lceil -\frac{(2-k)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} + h(c+1)(r_q s_d \alpha - r_d s_d) + r_q s_q \frac{e_{tr} + (c+1) e_{re}}{e_{tr} + e_{re}}}{k(r_d s_d - r_q s_d \alpha)} \right\rceil \tag{2.2.30}$$

è il primo m per cui l'Equazione 2.2.28 diventa positiva. É dunque possibile restringere la ricerca di x^* a $(m-1)(k-1) \leq x^* \leq m(k-1)$ utilizzando così solo tempo $O(\log k)$ per la ricerca binaria.

2.2.5.2 Seconda parte

Dobbiamo ora investigare il secondo caso, cioè $x + 1 < k$. Notiamo che in questo caso tutta la dorsale fino a x sarà nera e dovremo disporre i restanti $k - x - 1$ nodi neri tra le foglie di

$Cat(0, x-1)$. Potremo usare eventualmente meno di k nodi neri, ma avremo sempre almeno una foglia nera.

Ridefiniamo $E'(x)$ come il costo minimo dell'albero quando x è l'ultimo nodo nero e vale $x < k-1$. Possiamo ottenere il costo $E'(x)$ sommando i costi:

- $\sum_{i=x}^{h}[cr_d s_d + r_d s_d(1+c)(i-x)]$: risalita fino a x delle informazioni generate dai nodi in $Cat(x,h)$,
- $x\sum_{i=x}^{h}(c+1)\alpha r_q s_d$: la risalita di questi stessi dati da x fino alla radice,
- $\sum_{i=1}^{x-1} i r_q s_d \alpha$: la risalita fino alla radice dei dati generati dai nodi $[0,\ldots,x-1]$,
- $\sum_{i=1}^{x-1} i c r_q s_d \alpha$: risalita delle informazioni generate dalle foglie di $Cat(0,x-1)$ contando la sola porzione di percorso dal padre delle foglie fino alla radice,
- $A(x)$: ovvero l'invio delle informazioni dalle foglie in $Cat(0,x-1)$ al loro padre,
- $x r_q s_q \frac{e_{tr}+(c+1)e_{re}}{e_{tr}+e_{re}}$: distribuzione dell'interrogazione.

Riassumendo abbiamo:

$$\begin{aligned}
E'(x) =& \sum_{i=x}^{h}[cr_d s_d + r_d s_d(1+c)(i-x)] + x\sum_{i=x}^{h}(c+1)\alpha r_q s_d \\
&+ \sum_{i=1}^{x-1} i r_q s_d \alpha + \sum_{i=1}^{x-1} i c r_q s_d \alpha + A(x) \\
&+ x r_q s_q \frac{e_{tr}+(c+1)e_{re}}{e_{tr}+e_{re}} \\
=& \sum_{i=0}^{h-x} cr_d s_d + \sum_{i=0}^{h-x}[r_d s_d(1+c)i] + x\sum_{i=0}^{h-x}(c+1)\alpha r_q s_d \\
&+ \sum_{i=1}^{x-1} i(c+1) r_q s_d \alpha + A(x) \\
&+ x r_q s_q \frac{e_{tr}+(c+1)e_{re}}{e_{tr}+e_{re}} \\
=&(h-x+1)cr_d s_d + \frac{(h-x)(h-x+1)}{2} r_d s_d(1+c) + x(h-x+1)(c+1)\alpha r_q s_d \\
&+ \frac{x(x-1)}{2}(c+1) r_q s_d \alpha + A(x) \\
&+ x r_q s_q \frac{e_{tr}+(c+1)e_{re}}{e_{tr}+e_{re}}
\end{aligned} \tag{2.2.31}$$

Le cx foglie in $Cat(0,x-1)$ inviano informazioni ai loro padri per un costo totale di $A(x)$. Tali foglie ricevono tutte l'interrogazione, dunque avremo un guadagno ponendone nere il massimo numero possibile. Tra esse $k-x-1$ saranno dunque nere mentre le restanti $cx-(k-x-i)$ rimarranno bianche. Le foglie sono tra loro indistinguibili, non importa dunque quali renderemo effettivamente nere. Osservando che il numero di foglie cx potrebbe essere minore dei nodi neri a disposizione, possiamo definire $A(x)$ in questo modo:

$$A(x) = \begin{cases} cx\alpha r_q s_d & \text{se } cx \leq k-x-1 \\ (k-x-1)\alpha r_q s_d + [cx-(k-x-1)]r_d s_d & \text{altrimenti} \end{cases} \tag{2.2.32}$$

dove nel primo caso abbiamo tutte le foglie nere, mentre nel secondo solo $k-x-1$ sono nere. Sostituendo tale valore all'interno dell'Equazione 2.2.31 possiamo calcolare il valore $E'(x)$ in tempo $O(1)$.

2.2.5.3 Pseudocodice

Presentiamo ora lo pseudocodice per riassumere quanto detto.

Algoritmo 15: Modello 1, limitato, caterpillar regolare

$E = +\infty$;
// Caso $x+1 < k$
for $x = 0$ *to* $k-1$ **do**
  $E = \min\{E, E'(x)\}$;
end
$m = \left\lceil -\frac{(2-k)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} + h(c+1)(r_q s_d \alpha - r_d s_d) + r_q s_q \frac{e_{tr}+(c+1)e_{re}}{e_{tr}+e_{re}}}{k(r_d s_d - r_q s_d \alpha)} \right\rceil$;
// Caso $x+1 \geq k$
for x *where* $(m-1)(k-1) \leq x \leq m(k-1)$ **do** //Con ricerca binaria
  $E = \min\{E, E(x)\}$;
end

Al termine della computazione E contiene il costo ottimo dell'albero. Il primo ciclo è svolto $O(k)$ volte a tempo costante, mentre la ricerca binaria effettua $O(\log k)$ cicli ognuno a tempo costante. Per piazzare i nodi neri al termine della computazione è comunque necessario tempo $O(k)$. Si noti tuttavia che nel caso $k > h$ il primo ciclo effettua solo $O(h)$ iterazioni, mentre la ricerca binaria nel secondo caso controlla al più $\log h$ elementi. Il costo dell'algoritmo dunque diventa dunque $O(\min\{k, h\})$. Per ricostruire la colorazione ottima occorre tenere traccia di quale x^* minimizza E. Se x^* viene trovato nel primo ciclo, allora occorre porre neri tutti i nodi $[0, x^*]$ e porre nere tutte le foglie in $Cat(0, x^* - 1)$ fino ad esaurire i nodi neri. Se invece x^* viene individuato nel secondo ciclo, occorre calcolare la posizione nei nodi neri calcolando σ_j^* tramite l'Equazione 2.2.26 sostituendo x con x^*.

Capitolo 3

Modello 2

Vogliamo ora considerare un nuovo modello di comunicazione raffinando il precedente, in particolare permettiamo che un messaggio di interrogazione, possa essere inviato dal padre ad un sottoinsieme dei suoi figli. L'invio delle informazioni avviene sempre in broadcast ma i figli possono decidere se ricevere o meno l'interrogazione. Il costo di invio dei dati definito in 1.1.1 è dunque

$$\begin{cases} 1 & \text{se } j \text{ è padre di } i \\ b_i = \frac{e_{tr}+e_{re}r}{e_{tr}+e_{re}} & \text{altrimenti} \end{cases} \tag{3.0.1}$$

dove b_i è il costo di invio delll'interrogazione, ed r è il numero di figli di i neri o con discendenti neri (che dunque devono ricevere l'interrogazione).

Come nel primo modello, vale ancora il Corollario 2.0.6 (se $\alpha r_q \geq r_d$ allora ogni nodo sarà bianco), *quindi tutti gli algoritmi che presenteremo assumeranno di essere nel caso* $\alpha r_q < r_d$. Anche qui ricordiamo che non è vero il contrario, cioè la soluzione ottima potrebbe avere solo nodi neri (con radice sempre nera), ma essere nel caso $\alpha r_q < r_d$.

Per semplicità di trattazione, ove non diversamente segnalato, assumeremo che il costo di ricezione dell'interrogazione $\frac{e_{re}}{e_{tr}+e_{re}}$ sia assegnato al nodo ricevente, e che il il solo costo di invio dell'interrogazione $\frac{e_{tr}}{e_{tr}+e_{re}}$ sia attribuito al mittente.

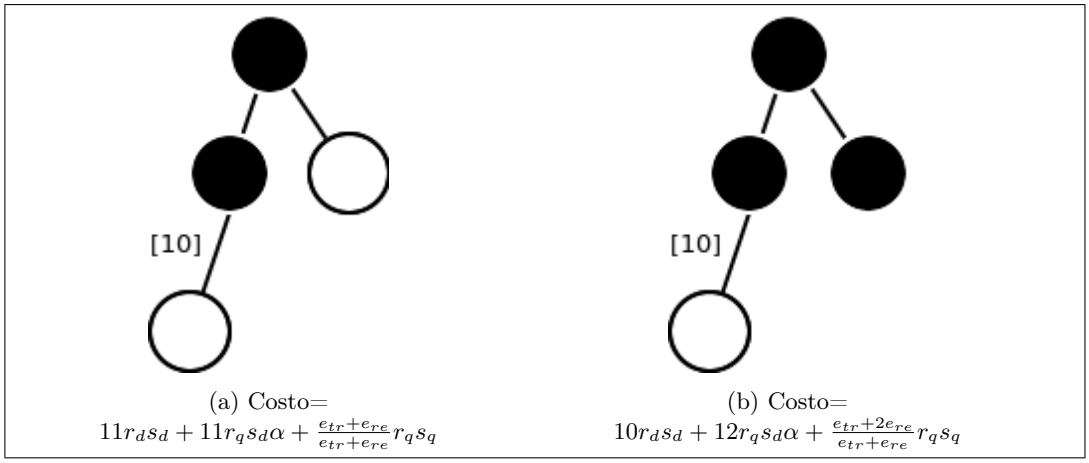
3.1 Non limitato

Studiamo prima il caso non limitato, cioè supponiamo di avere a disposizione un qualsiasi numero di nodi neri. Il Teorema 2.1.1 degli antenati (un nodo nero ha tutti gli antenati neri) è applicabile anche al questo modello, poiché tutti gli antenati di un nodo nero ricevono l'interrogazione. Quindi

Teorema 3.1.1. ***Teorema degli antenati***
Nella colorazione ottima di un albero, se i è un nodo nero, allora esso avrà tutti gli antenati neri.

da questo si evince anche che un nodo bianco avrà solo discendenti bianchi.

Differentemente dal primo modello, non vale il Teorema 2.1.2 dei fratelli cioè non è vero che un nodo nero avrà tutti i fratelli neri, poiché questi potrebbero non ricevere l'interrogazione. Possiamo facilmente verificare che inviare l'interrogazione a tutti i figli non è sempre la scelta migliore, tale osservazione è mostrata nella Figura 3.1.1 . È dunque possibile che nodi fratelli abbiano colorazione diversa.



(a) Costo= $11r_d s_d + 11r_q s_d \alpha + \frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} r_q s_q$

(b) Costo= $10r_d s_d + 12r_q s_d \alpha + \frac{e_{tr}+2e_{re}}{e_{tr}+e_{re}} r_q s_q$

Figura 3.1.1: Controesempio che dimostra come inviare l'interrogazione a tutti i figli e renderli neri non sia sempre la soluzione ottima. L'etichetta [10] indica che il nodo ha 10 figli. Esistono dei valori per le variabili che rendono il costo di (a) inferiore a quello di (b), per esempio l'assegnamento $r_d = s_d = r_q = s_q = 1, \alpha = 1/2, e_{re} = 4, e_{tr} = 1$ rispetta tale proprietà. Si può inoltre verificare esaustivamente che quella in (a) è la colorazione ottima.

La frontiera in questo caso sarà diversa da quella esposta nel primo modello, poiché padre e figlio possono appartenervi entrambi. Enunciamo dunque:

Teorema 3.1.2. ***Teorema della frontiera***
La soluzione ottima nel caso non limitato è data da un insieme di nodi neri detto ***Frontiera****, che hanno solo antenati neri e almeno un figlio bianco (o nesssun figlio).*

che si ottiene banalmente per composizione dei teoremi prededenti.

Le soluzioni avranno dunque forma simile a quella mostrata in Figura 3.1.2, si noti la differenza con la Figura 2.1.2 .

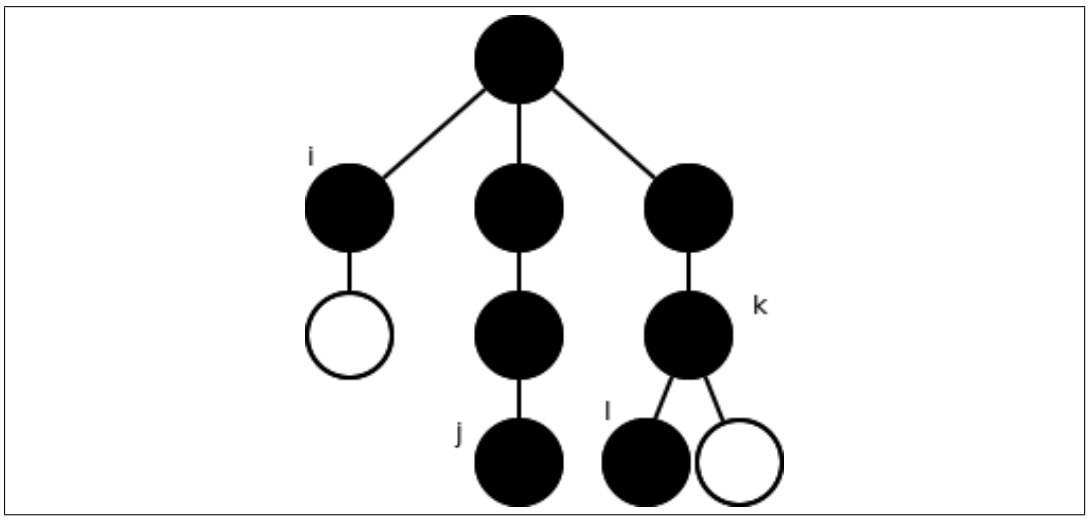


Figura 3.1.2: Esempio di soluzione nel caso non limitato. La frontiera che separa i nodi bianchi dai neri è composta dai nodi $\{i, j, k, l\}$

3.1.1 Albero

Presentiamo ora un algoritmo, diverso dai precedenti, per risolvere il problema nel caso in cui il grafo in input sia un albero generico. Definiamo come segue il costo $E(i)$:

Definizione 3.1.3. $E(i)$ costo ottimo del sottoalbero T_i, con un qualsiasi numero di neri, che comprende il costo di risalita dalle foglie fino ad i, il costo di risalita da i al padre di i e le eventuali ricezione ed inoltro dell'interrogazione da parte di i.

Diversamente dai precedenti algoritmi, qui il nodo che riceve l'interrogazione ne paga il costo di ricezione $\frac{e_{re}}{e_{re}+e_{tr}}$ mentre chia la invia paga il solo costo di invio $\frac{e_{tr}}{e_{re}+e_{tr}}$. Ci occorre anche definire:

Definizione 3.1.4. $E_w(i)$ costo dell'albero T_i con soli nodi bianchi, compreso i bianco, comprenivo anche del costo di risalita da i verso il padre di i.

Il programma usa la seguente relazione di programmazione dinamica:

$$E(i) = \min \begin{cases} r_q \alpha s_d \left|T_i\right| + \frac{e_{re}}{e_{re}+e_{tr}} r_q s_q + \frac{e_{tr}}{e_{re}+e_{tr}} r_q s_q + \sum_{j \in C_i} E(j) \\ r_q \alpha s_d \left|T_i\right| + \frac{e_{re}}{e_{re}+e_{tr}} r_q s_q + \sum_{j \in C_i} E_w(j) \\ E_{w(i)} = r_d s_d \left|T_i\right| + \sum_{j \in C_i} E_w(j) \end{cases} \tag{3.1.1}$$

Per ogni i non foglia non radice.

$$E(i) = \min \left\{ r_d s_d, r_q \alpha s_d + \frac{e_{re}}{e_{re}+e_{tr}} r_q s_q \right\} \tag{3.1.2}$$

se i è una foglia

Il secondo caso dell'Equazione 3.1.1 rappresenta il costo in cui i è nero e tutti i suoi discendenti sono bianchi. In questo caso il costo è ricavato sommando l'invio dei dati generati di $|T_i|$ nodi da i al padre, sommando la ricezione dell'interrogazione da parte di i, e sommando i costi dei sottoalberi completamente bianchi T_j. Il tezo caso dell'Equazione 3.1.1 invece

rappresenta il costo di T_I quando contiene solo nodi bianchi e si ottiene sommando al costo di invio dei dati da i al padre, il costo dei sottoalberi T_j. Il primo caso rappresenta il costo di T_i quando i è nero, ed alcuni dei suoi sottoalberi hanno un qualsiasi numero di nodi neri (tali sottoalberi avranno anche loro internamente una frontiera). Il costo qui è dato sommando l'invio di dati da i al padre, sommando la ricezione e l'inoltro dell'interrogazione da parte di i, e sommando il costo dei sottoalberi T_j.

Assumiamo che i costi del secondo e terzo caso siano corretti e ben definiti enunciamdo i seguenti lemmi:

Lemma 3.1.5. *Il secondo branch dell'Equazione 3.1.1 rappresenta il costo di T_i dove i è nero e tutti i suoi discendenti sono bianchi.*

Lemma 3.1.6. *Il terzo caso dell'Equazione 3.1.1 rappresenta il costo dell'albero T_i dove tutti i nodi, i compreso, sono bianchi.*

giustificheremo meglio il costo del primo caso in seguito.

L'algoritmo opera calcolando $E(i)$ dalle foglie fino ai figli della radice, in post-ordine. Inizialmente colora tutti i nodi di nero, successivamente trasforma in bianchi interi sottoalberi (ricordiamo infatti che secondo il Teorema 3.1.1 degli antenati i nodi bianchi hanno solo discendenti bianchi). Infine calcola il costo ottimo E dell'albero in questo modo:

$$E = \min \left\{ \sum_{j \in C_i} E_w(j), \sum_{j \in C_i} E(j) + \frac{e_{tr}}{e_{re} + e_{tr}} r_q s_q \right\} \tag{3.1.3}$$

Dove il primo caso rappresenta il costo dell'albero interamente bianco, ed il secondo rappresenta il costo con altri nodi neri. In E non è presente il costo di fuoriuscita di informazioni dalla radice verso il padre fittizio.
Questo è il costo ottimo E dell'albero con al più k nodi neri che cerchiamo.

3.1.1.1 Pseudocodice

Possiamo ora mostrare lo pseudocodice dell'algoritmo.

Algoritmo 16: Modello 2, non limitato, albero generico

```
1  foreach foglia i do
2  |  Poni i nero;
3  |  cost_1 = (e_re / (e_re + e_tr)) r_q s_q + r_q s_d α;
4  |  cost_2 = r_d s_d;
5  |  E(i) = min{cost_1, cost_2};
6  |  E_w(i) = cost_2;
7  |  if cost_2 ≤ cost_1 then
8  |  |  Poni i bianco;
9  |  end
10 end
11 foreach nodo i non foglia, non radice in postordine do
12 |  Poni i nero;
13 |  cost_1 = r_q α s_d |T_i| + (e_re / (e_re + e_tr)) r_q s_q + (e_tr / (e_re + e_tr)) r_q s_q + Σ_{j∈C_i} E(j);
14 |  cost_2 = r_q α s_d |T_i| + (e_re / (e_re + e_tr)) r_q s_q + Σ_{j∈C_i} E_w(j);
15 |  cost_3 = E_{w(i)} = r_d s_d |T_i| + Σ_{j∈C_i} E_w(j);
16 |  E_w(i) = cost_3;
17 |  E(i) = min {cost_1, cost_2, cost_3};
18 |  if cost_3 = min {cost_1, cost_2, cost_3} then
19 |  |  Poni i e tutti i suoi discendenti bianchi;
20 |  if cost_2 = min {cost_1, cost_2, cost_3} then
21 |  |  Poni tutti i discendenti di i bianchi;
22 |  end
23 end
24 Poni la radice nera;
25 cost_1 = (e_tr / (e_re + e_tr)) r_q s_q + Σ_{j∈C_i} E(j);
   // Caso tutto bianco
26 cost_2 = Σ_{j∈C_i} E_w(j);
27 E = min {cost_1, cost_2};
28 if cost_2 ≤ min {cost_1, cost_2} then
29 |  Poni bianchi tutti i nodi non radice;
30 end
```

Il secondo ciclo **foreach** viene eseguito n volte, ed ogni iterazione impiega tempo $O(c)$, dove c è il massimo numero di figli dell'albero (dovuto alle sommatorie). Il costo dell'algoritmo sembrerebbe dunque $O(nc)$, ma ora mostreremo come abbassarlo a $O(n)$.

Creiamo delle variabili ausiliare in modo da poterle richiamare direttamente nel calcolo di $E(i)$. Siano

$$L(i) = \sum_{j \in c_i} E(j)$$

e

$$L_w(i) = \sum_{j \in c_i} E_w(j)$$

e sia i' il nodo padre di i. Tali variabili devono essere tutte inizializzate a zero. Ad ogni iterazione del ciclo più esterno, aggiungiamo il valore $E(i)$ calcolato nell'iterazione corrente

alla variabile $L(i')$, ed il valore appena calcolato $E_w(i)$ a $L_w(i')$. Possiamo così definire l'algoritmo ottimizzato:

Algoritmo 17: Modello 2, non limitato, albero generico, ottimizzato

```
   // Inizializzazione
 1 Set L(i) = 0, L_w(i) = 0, ∀i;
 2 foreach leaf i do
 3 |   Poni i nero;
 4 |   cost_1 = e_re/(e_re+e_tr) r_q s_q + r_q s_d α;
 5 |   cost_2 = r_d s_d;
 6 |   E(i) = min{cost_1, cost_2};
 7 |   E_w(i) = cost_2;
 8 |   L(i') = L(i') + E(i);
 9 |   L_w(i') = L_w(i') + E_w(i);
10 |   if cost_2 ≤ cost_1 then
11 |   |   Poni i Bianco;
12 |   end
13 end
14 foreach nodo i non foglia, non radice in postordino do
15 |   Poni i nero;
16 |   cost_1 = r_q α s_d |T_i| + e_re/(e_re+e_tr) r_q s_q + e_tr/(e_re+e_tr) r_q s_q + L(i);
17 |   cost_2 = r_q α s_d |T_i| + e_re/(e_re+e_tr) r_q s_q + L_w(i);
18 |   cost_3 = E_w(i) = r_d s_d |T_i| + L_w(i);
19 |   E_w(i) = cost_3;
20 |   E(i) = min{cost_1, cost_2, cost_3};
21 |   if i' ≠ radice then
22 |   |   L(i') = L(i') + E(i);
23 |   |   L_w(i') = L_w(i') + E_w(i);
24 |   if cost_3 = min{cost_1, cost_2, cost_3} then
25 |   |   Poni i e tutti i suoi discendenti bianchi;
26 |   if cost_2 = min{cost_1, cost_2, cost_3} then
27 |   |   Poni bianchi tutti i discendenti di i;
28 |   end
29 end
30 Poni la radice nera;
31 cost_1 = e_tr/(e_re+e_tr) r_q s_q + Σ_{j∈C_i} E(j);
   // Caso tutto bianco
32 cost_2 = Σ_{j∈C_i} E_w(j);
33 E = min{cost_1, cost_2};
34 if cost_2 ≤ min{cost_1, cost_2} then
35 |   Poni bianchi tutti i nodi non radice;
36 end
```

Il costo dell'algoritmo diventa quindi $O(n)$. Per ottenere la colorazione ottima è sufficiente tenere traccia delle trasformazioni di colore che avvengono nell'algoritmo. poiché ogni nodo viene sbiancato al più una sola volta è possibile ottenere la colorazione ottima senza costi aggiuntivi.

3.1.1.2 Dimostrazione di correttezza ed ottimalità

Dimostrazione. Nella prima parte si dimostra per induzione strutturale che $E(i)$, è ben definito ed ottimo per ogni i.
Nella seconda parte dimostreremo che E è ben definito avendo precalcolato i valori $E(i)$ dei figli della radice.

Prima parte Dimostrare che $E(i)$ sia ben definito ed ottimo per induzione strutturale.

Se i é foglia allora è banalmente vero. Se i non è foglia procediamo per casi sulla colorazione ottima dell'albero T_i.

Caso 1. Nella colorazione ottima Z_i dell'albero T_i, i è nero ed ha almeno un figlio nero. In questo caso le colorazioni ottime dei sottoalberi $T_j, j \in C_i$ (j figli di i) sono identiche a quelle nella colorazione Z_i. Se così non fosse, esisterebbe almeno un k, figlio di i, che ha una colorazione ottima diversa da quella contenuta in Z_i; potremmo allora sostituire [1] tale colorazione dell'albero T_k in Z_i, ed ottenere così una nuova colorazione $Z_i^{'}$ con costo minore di Z_i, ma ciò è assurdo poiché Z_i era la colorazione dell'ottimo. Essendo dunque ottime le colorazioni dei sottoalberi T_j in Z_i, il loro costo è proprio $E(j)$ per ipotesi induttiva.
Avendo ora tutti i costi $E(j)$ si può calcolare $E(i)$ secondo la colorazione Z_i sommando i costi:

- $\sum_{j \in C_i} E(j)$: costo dei sottoalberi,
- $r_q \alpha s_d |T_i|$: costo di risalita da i al padre di i delle informazioni generate in $|T_i|$,
- $\frac{e_{tr}}{e_{re}+e_{tr}} r_q s_q$: invio dell'interrogazione ai figli j (poiché in Z_i i ha almeno un figlio nero),
- $\frac{e_{re}}{e_{re}+e_{tr}} r_q s_q$: ricezione dell'interrogazione da parte di i.

Riassumendo quindi

$$E(i) = r_q \alpha s_d |T_i| + \frac{e_{re}}{e_{re}+e_{tr}} r_q s_q + \frac{e_{tr}}{e_{re}+e_{tr}} r_q s_q + \sum_{j \in C_i} E(j)$$

che corrisponde al primo caso dell'Equazione (3.1.1). Fino a qui abbiamo mostrato come calcolare il costo $E(i)$ nel primo caso, ma dobbiamo anche essere sicuri che l'Equazione 3.1.1 scelga il primo branch e non un altro. Di questo siamo sicuri, poiché il secondo ed il terzo branch esprimono il costo dell'albero T_i con una diversa colorazione, come mostrato nei Lemmi 3.1.5 e 3.1.6.

Caso 2. La colorazione ottima Z_i di T_i, è composta da i nero e tutti i suoi discendenti bianchi. In questo caso il costo corretto per $E(i)$ deve essere calcolato con la formula nel secondo branch dell'Equazione 3.1.1. Dobbiamo essere sicuri che se Z_i rientra in questo caso allora l'algoritmo secondo l'Equazione 3.1.1 non scelga uno degli altri branch. L'algoritmo non sceglierà mai il terzo branch poiché

[1] Eventualmente tale sostituzione potrebbe fare in modo che tutti i discendenti di i siano bianchi, ed occorra eliminare il costo di invio dell'interrogazione da parte di i; in questo caso comunque il costo dell'albero T_i scenderebbe, facendo rimanere invariata la veridicità dell'affermazione.

esso rappresenta il costo dell'albero T_i con una colorazione diversa da Z_i. L'algoritmo inoltre non sceglierà mai il primo caso dell'Equazione 3.1.1 poiché questo rappresenta il costo di una colorazione S_I diversa da Z_i [2].

Caso 3. La colorazione ottima Z_i di T_i, ha tutti i nodi bianchi, compreso i. Il costo di questo caso si calcola con la formula che appare nel terzo branch dell'Equazione 3.1.1. L'algoritmo arrivato a questo punto non sceglierè mai il secondo branch dell'Equazione 3.1.1, poiché esso rappresenta il costo con una diversa colorazione come nel secondo branch dell'Equazione 3.1.5, che deve essere per forza maggiore. L'algoritmo inoltre non sceglierà mai il primo branch poiché esso rappresenta il costo di una diversa colorazione (con i nero) che deve quindi essere maggiore del costo di Z_i.

Seconda parte Verifichiamo che sia corretta l'Equazione 3.1.3. Procediamo per casi ricordando che ora i=0=radice, e quindi i è sempre nera.

Caso 1. La colorazione ottima Z dell'albero contiene dei neri sotto la radice. In questo caso le colorazioni ottime dei sottoalberi T_j figli della radice, sono identiche a quelle nelle colorazione Z. Se così non fosse potremmo sostituire la colorazione ottima di un sottoalbero T_j in Z ottenendo una nuova colorazione $Z^{'}$ con costo inferiore al costo di Z, ma ciò è assurdo poiché Z è la colorazione ottima. Avendo quindi i corretti $E(j)$ per tutti i figli j della radice, possiamo calcolare E con l'Equazione 3.1.3.

Caso 2. La colorazione ottima è composta da radice nera e da tutti gli altri nodi bianchi. In questo caso il costo corretto è da calcolarsi con il primo caso dell'Equazione 3.1.3. Potrebbe l'algoritmo scegliere, erroneamente, il secondo caso? No, poiché esso rappresenta il costo di una diversa colorazione.

□

[2] S_i avrà i nero, e tutti i sottoalberi T_j con una qualche colorazione. In ogni caso questa formula calcola correttamente il costo dell'albero T_i con la colorazione S_i. Nel caso limite in cui tutti i sottoalberi $T_j, j \in c_i$, in S_i abbiamo una colorazione completamente bianca allora il primo branch avrà costo maggiore o uguale al secondo, di un fattore $\frac{e_{tr}}{e_{re}+e_{tr}} r_q s_q$.

3.1.2 Albero regolare completo

Supponiamo ora di avere un albero regolare completo, dove ogni nodo ha esattamente c figli e tutte le foglie sono allo stesso livello; richiamiamo dunque quanto detto nel primo modello nella Sezione 2.1.2.

Per poter utilizzare l'Algoritmo 3 in questo caso, dobbiamo prima dimostrare che la frontiera è composta da nodi alla medesima profondità, e quindi la soluzione apparirà come in Figura 2.1.3 .

Dimostrazione. Per $c = 1$ ci si riconduce ad un cammino e la proprietà è banalmente vera. Ricordiamo che in un albero regolare completo due sottoalberi radicati alla stessa profondità hanno la medesima topologia. Vogliamo mostrare che la colorazione ottima Z dell'albero ha la frontiera tutta allo stesso livello. Supponiamo per assurdo che la colorazione Z' sia ottima ed abbia una frontiera con nodi a livello diverso [3]. Siano allora T_i e T_j i due sottoalberi con colore della radice diverso, radicati allo stesso livello e con profondità della radice minima. Così scelti, i padri di i e di j saranno neri. Calcoliamo il costo di un nodo $e(i)$ in modo simile a come definito in 2.2.1 a pagina 34 ma non potendo definire il costo di distribuzione dell'interrogazione in funzione dela profondità della frontiera, aggiungiamo i costi di ricezione ed invio dell'interrogazione ad ogni nodo. Per cui

$$e(i) = \begin{cases} d_i r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q & \text{se } i \text{ nero e non ha discendenti neri} \\ d_i r_q s_d \alpha + \frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} r_q s_q & \text{se } i \text{ nero ed ha discendenti neri} \\ (l+1) r_d s_d + (d_i - l - 1) r_q s_d \alpha & \text{se } i \text{ bianco} \end{cases}$$

non è mai possibile che un nodo bianco abbia disendenti neri. Sia inoltre il costo di un sottoalbero $E(i) = \sum_{j \in T_i} e(j)$ ed il costo totale dell'albero $E = \sum_{i \in T} e(i)$. Possiamo applicare iterativamente il seguente procedimento fintanto che rientriamo nel primo caso, o fino a quando non rientriamo nel secondo caso:

Caso 1. Se $E(i) = E(j)$ allora possiamo colorare T_j come T_i ed ottenere una nuova colorazione con costo identico, a cui riapplichiamo questo procedimento scegliendo i due nuovi sottoalberi T_i e T_j.

Caso 2. Se $E(i) < E(j)$ allora coloriamo T_j come T_i ottenendo una nuova colorazione Z'' con costo minore di Z', ma ciò è assurdo perché abbiamo supposto Z' ottima. Durante questa sostituzione i padri di i e j sono sempre neri, dunque i valori di l nei nodi di T_i e T_j non dipendono daigli antenati di i e j.

Applicando più volte questo metodo, se cadiamo nel secondo caso allora abbiamo dimostrato l'assurdo. Se rimaniamo nel primo caso, cioè esistono sempre T_i e T_j con costo uguale e colorazione diversa, allora continuiamo a cambiare la colorazione fino a trovarne una nuova che avrà la frontiera tutta allo stesso livello e costo identico a Z'. □

Questa dimostrazione è sufficiente per poter applicare l'Algoritmo 3 al caso corrente.

[3] In Z' ogni nero avrà tutti gli antenati neri per il teorema degli antenati 3.1.1 a pagina 64, altrimenti Z' non sarebbe ottimo banalmente.

3.1.3 Cammino

Supponiamo ora che l'albero in input sia un cammino di n nodi dove ogni nodo ha esattamente 1 figlio. Per il Teorema 3.1.1 degli antenati la frontiera sarà composta da un solo nodo, con tutti i discendenti bianchi e tutti gli antenati neri; la soluzione ottima apparirà dunque come in figura 2.1.4 a pagina 18.

Poiché ogni nodo ha un solo figlio, quando l'interrogazione viene inviata questa viene ricevuta da tutti i figli, ci possiamo ricondurre dunque così al primo modello. Possiamo così applicare a questo caso quanto esposto nella Sezione 2.1.3 ed utilizzare per la risoluzione l'Algoritmo 4 a pagina 20. Ricordiamo che tale algoritmo trova il nodo di frontiera in tempo $O(i)$ e colora i nodi neri in tempo $O(k_u)$. Il costo energetico della soluzione ottima si ottiene invece in tempo costante.

3.1.4 Caterpillar

Studiamo ora il caso in cui l'albero sia un caterpillar generico. Osserviamo innanzitutto che il nodo nero a maggior profondità q sarà nella dorsale. Se così non fosse potremo spostare tale nero nel fratello in dorsale senza che il costo energetico totale aumenti. Il nodo q avrà tutti gli antenati neri per il Teorema 3.1.1 degli antenati. Diversamente da quanto studiato nel primo modello, le foglie in $Cat(0, q-1)$ non saranno banalmente nere, ma è possibile individuare una condizione per cui saranno tutte nere o tutte bianche.

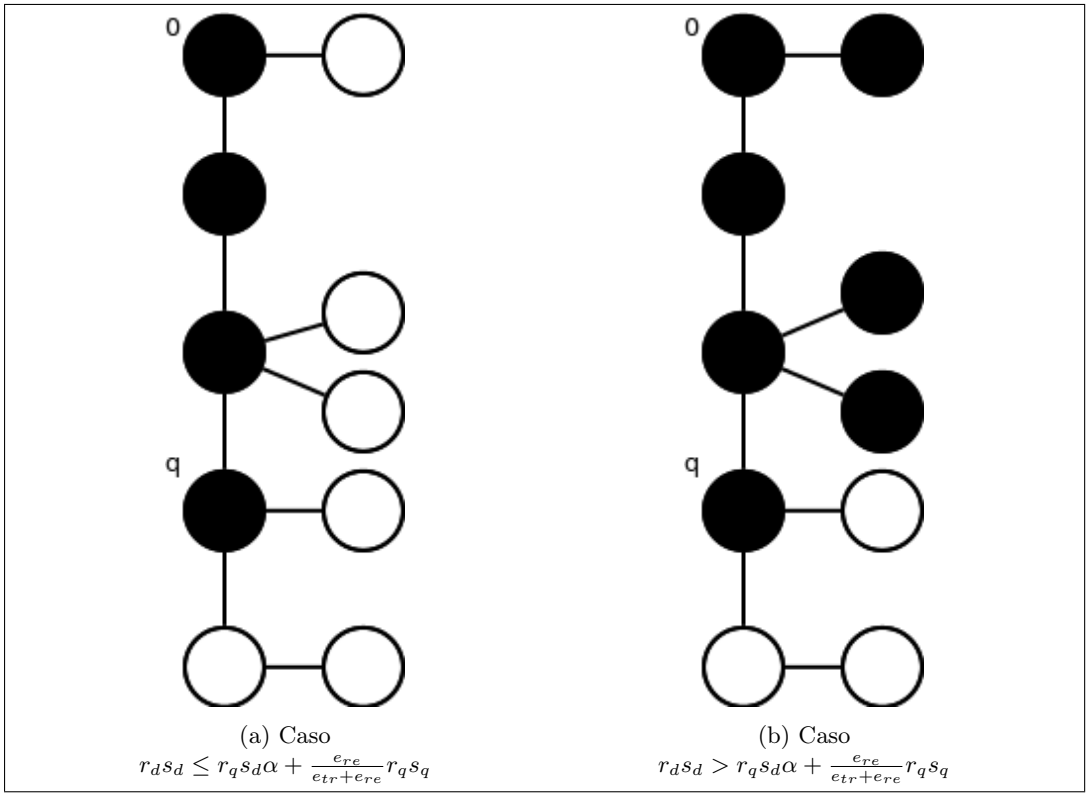
Teorema 3.1.7. *Se* $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$ *allora nella soluzione ottima tutte le foglie in* $Cat(0, q-1)$ *sono bianche, altrimenti sono tutte nere.*

Dimostrazione. Si prendano due caterpillar identici, con i nodi $[0, \ldots, q]$ neri e tutti i successori di q bianchi, che differiscono solo per il colore di una foglia in $Cat(0, q-1)$. La differenza di costo totale tra i due sarà di

$$r_q s_d \alpha + \frac{e_{re}}{e_{tr} + e_{re}} r_q s_q - r_d s_d$$

poiché questo costo non dipende dal colore delle altre foglie. Dunque quando $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$ avrà costo minore il caterpillar con la foglia in questione bianca; viceversa nel caso opposto mettere nera la foglia abbassa il costo. □

Quindi fissato q la soluzione ottima può essere solo una tra le due proposte in Figura 3.1.3.



(a) Caso $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$

(b) Caso $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$

Figura 3.1.3: La soluzione ottima è una delle due presentate.

Secondo questa definizione avremo quindi

$$k_u = \begin{cases} q+1 & \text{se } r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q \\ N'_{q-1}+1 & \text{se } r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q \end{cases}$$

Osserviamo che per $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$ la soluzione sarà come quella attesa su caterpillar nel primo modello, già studiato nella Sezione 2.1.5. Questo caso può dunque essere risolto mediante l'Algoritmo 9 in tempo $O(h)$. La soluzione del problema nel primo modello rappresenta quindi un lowerbound di costo per il caso di un caterpillar non limitato nel secondo modello.
Proseguiamo la nostra trattazione investigando il solo caso $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$ in cui le foglie sono tutte bianche.

Sia $E(q)$ il costo energetico dell'albero quando l'ultimo nero è q, non comprensivo del costo di distribuzione dell'interrogazione. È possibile calcolare il costo totale dell'albero E mediante:

$$E = \min_{0 \leq q \leq h} \left\{ E(q) + q \frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} r_q s_q \right\} = \min_{0 \leq q \leq h} \{ E(q) + q r_q s_q \} \tag{3.1.4}$$

Sia $F(0,b) = N'_b - b - 1$ il numero di foglie in $Cat(0,b)$. Possiamo calcolare $E(q)$ sommando:

- $M_q r_d s_d$: costo dei messaggi scambiati in $Cat(q,h)$,
- $N_q q r_q s_d \alpha$: costo della risalita fino alla radice delle informazioni generate da $Cat(q,h)$,
- $F(0,q-1) r_d s_d$: invio al padre di messaggi grezzi da parte delle foglie in $Cat(0,q-i)$,
- $[M'_{q-1} - F(0,q-1)] r_q s_d \alpha$: trasmissione fino alla radice delle informazioni generate dai nodi in $Cat(0,q-1)$ tramite messaggi di risposta.

Riassumendo:

$$\begin{aligned} E(q) =& M_q r_d s_d + N_q q r_q s_d \alpha + F(0,q-1) r_d s_d + [M'_{q-1} - F(0,q-1)] r_q s_d \alpha = \\ =& [M_q + F(0,q-1)] r_d s_d + [N_q q + M'_{q-1} - F(0,q-1)] r_q s_d \alpha \end{aligned} \tag{3.1.5}$$

i valori $E(q)$ si possono calcolare in tempo costante avendo calcolato a priori i valori M_q, N_q, N'_q e M'_q tramite gli Algoritmi 5, 6, 7 e 8 in tempo $O(h)$.

L'algoritmo cerca esaustivamente quale sia l'ultimo nodo nero q che minimizza E calcolando $E(q)$ ad ogni iterazione.

Possiamo chiederci se valga la proprietà bitonica su caterpillar, come esposta nella Definizione 2.1.11; se valesse potremmo trovare q ottimo in tempo $O(\log h)$ tramite ricerca binaria su sequenza bitonica. Se siamo nel caso $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$ allora tutte le foglie in $Cat(0,q-1)$ sono nere e rientriamo nel primo modello; putroppo in questo caso non vale la proprietà bitonica come mostrato in Figura 2.1.7.

3.1.4.1 Pseudocodice

L'algoritmo calcola inizialmente i valori $M_q, N_q, N'_q, M'_q, \forall q \in [0,h]$. Se rientriamo nel primo caso utilizziamo l'Algoritmo 9. Se invece rientriamo nel secondo caso, crea una tabella con

$h+1$ valori in cui salva i valori $E(q), \forall q \in [0,h]$. Cerca infine il minimo costo E tramite l'Equazione 3.1.4

Algoritmo 18: Modello 2, non limitato, caterpillar

```
1  Effettua precomputazioni;
2  if r_d s_d > r_q s_d α + e_re/(e_tr+e_re) r_q s_q then
3  |  Usa l'Algoritmo 9;
4  else
5  |  for q = 0 to h do
6  |  |  E[q] = [M_q + F(0, q−1)] r_d s_d + [N_q q + M'_{q−1} − F(0, q−1)] r_q s_d α;
7  |  end
8  |  E = +∞;
9  |  for q = 0 to h do
10 |  |  E_temp = E[q] + q r_q s_q;
11 |  |  if E_temp < E then
12 |  |  |  E = E_temp;
13 |  |  end
14 |  end
15 end
```

Al termine della computazione la variabile E contiene il costo minimo dell'albero. Il costo per le precomputazioni e per l'Algoritmo 9 è $O(h)$, i cicli *foreach* vengono eseguiti $O(h)$ volte ed usano tempo costante ad ogni iterazione. Dunque il costo dell'algoritmo è $O(h)$. Per individuare il q corretto, e quindi individuare quali nodi sono da annerire, è sufficiente tenere traccia del q che minimizza E. Per piazzarre effettivamente i nodi neri è però necessarrio tempo $O(k_u)$.

3.1.4.2 Correttezza ed ottimalità

Dopo aver trovato che le soluzioni possono essere in una delle forme di Figura 3.1.3, l'algoritmo effettua una ricerca esaustiva tra tutte quelle possibili. L'ottimalità è garantita dal fatto che viene scelta la soluzione di costo minimo. La dimostrazione di correttezza si deve dunque limitare a verificare che le Equazioni 3.1.4 e 3.1.5 siano ben definite nel caso $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$, cioè che calcolino correttamente i valori desiderati.

Il calcolo di E è corretto poiché ricerca il minimo fra tutti gli $E(q)$, che come abbiamo detto, comprendono tutti i costi ad eccezione della diffusione dell'interrogazione. Essendo q l'ultimo nodo nero, l'interrogazione deve essere inviata e ricevuta da esattamente q nodi, dunque il costo per diffondere l'interrogazione è esattamente $q\frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} r_q s_q$.

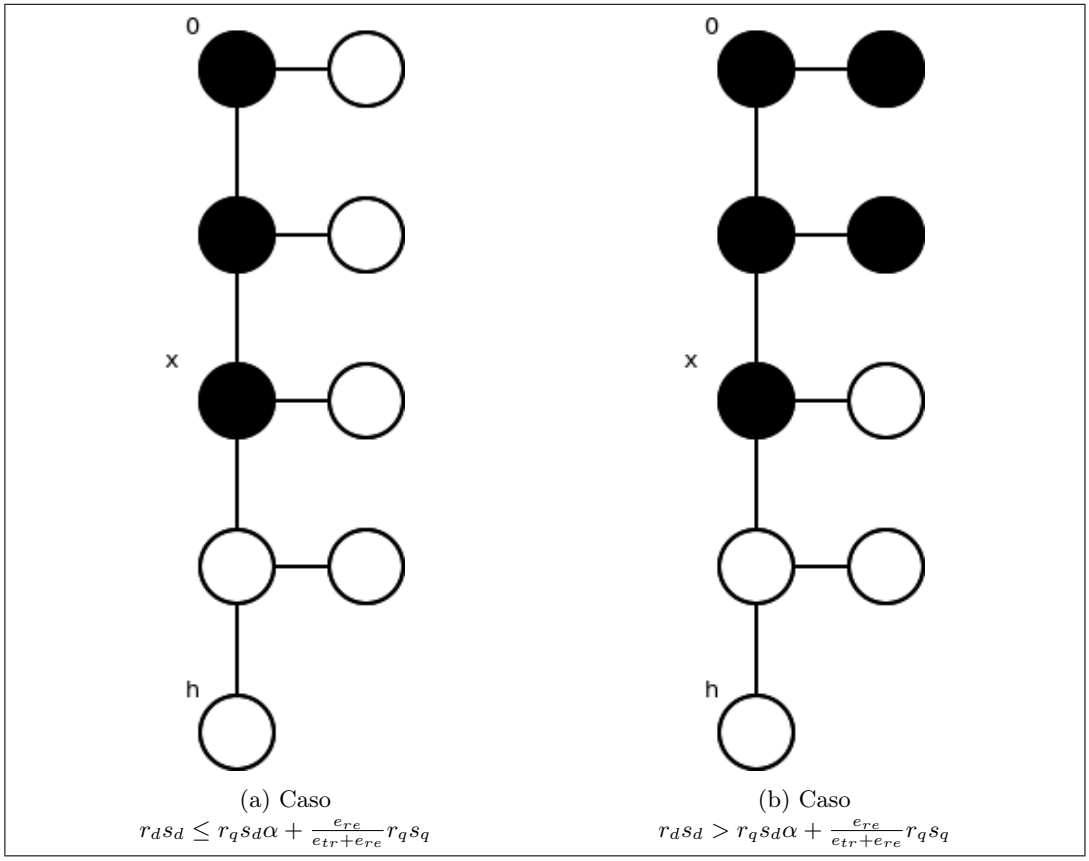
Per convicersi che il calcolo di $E(q)$ è corretto, verifichiamo che esso conteggi il numero esatto di messaggi scambiati M_0. Dall'Equazione 3.1.5 si vede che conteggia $M_q + F(0, q-1) + N_q q + M'_{q-1} - F(0, q-1)$ messaggi, e riscrivendo tale numero si ottiene proprio

$$\begin{aligned} & M_q + F(0,q-1) + N_q q + M'_{q-1} - F(0,q-1) = \\ =& M'_{q-1} + M_q + N_q q = \\ =& M_0 \end{aligned} \tag{3.1.6}$$

per l'Equazione 2.1.8.

3.1.5 Caterpillar regolare

Supponiamo ora che l'albero in input sia un caterpillar regolare dove ogni nodo della dorsale ha un numero costante c di figli, di cui k foglie, ad eccezione del nodo h che sarà sempre una foglia. Le proprietà che valgono su caterpillar generici viste nella Sezione 3.1.4 valgono ovviamente anche su caterpillar regolari, in particolare la soluzione ottima avrà tutti i nodi neri fino a q, i discendenti di q saranno bianchi, e le foglie di $Cat(0, x-1)$ saranno tutte bianche o tutte nere. In altre parole la soluzione appartiene ad uno dei due casi in Figura 3.1.4. Come visto nel Teorema 3.1.7 qui applicabile, la condizione che separa i due casi è $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$.



(a) Caso $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$

(b) Caso $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$

Figura 3.1.4: Caterpillar regolare con $c = 2, h = 4$: la soluzione ottima è una delle due presentate.

Quando ricadiamo nel caso $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$, allora la soluzione ottima è identica a quelle che si cercano su caterpillar regolare nel primo modello di comunicazione, poiché ogni nodo che invia l'interrogazione la invia a tutti i figli. È dunque possibile risolvere questo caso con l'Algoritmo 10 trattato nella Sezione 2.1.6. Essendo lo stesso problema sul primo modello un sottocaso, il costo di tale algoritmo ci darà un lowerbound per il caso corrente.

Studiamo ora il caso $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$. Poiché i nodi neri stanno solo sulla dorsale:

$$b_i = \frac{e_{tr} + e_{re}}{e_{tr} + e_{re}} = 1$$

Definiamo quindi

Definizione 3.1.8. Sia $E(x)$ costo dell'albero quando l'ultimo nero è x, tutti i nodi $[0, \dots, x]$ sono neri, tutti i nodi di $Cat(x+1, h)$ sono bianchi e tutte le foglie sono bianche.

Possiamo calcolare $E(x)$ sommando i costi di:

- $kxr_d s_d + \sum_{i=1}^{x-1} kir_q s_d \alpha$: risalita dei messaggi dalle foglie di $Cat(0, x-1)$ alla radice,
- $\sum_{i=1}^{x} r_q s_d \alpha$: trasporto fino alla radice dei dati generati dai nodi $[0, \dots, x]$,
- $\sum_{i=1}^{h-x} i(k+1) r_d s_d$: invio fino a x delle informazioni generate in $Cat(x, h)$,
- $(h-x)(k+1)xr_q s_d \alpha$: trasporto alla radice delle informazioni generate in $Cat(x, h)$,
- $xb_i r_q s_q = xr_q s_q$: costo di diffusione dell'interrogazione.

Ricomponendo le somme abbiamo:

$$\begin{aligned} E(x) =& kxr_d s_d + \sum_{i=1}^{x-1} kir_q s_d \alpha + \sum_{i=1}^{x} r_q s_d \alpha + \sum_{i=1}^{h-x} i(k+1) r_d s_d \\ &+ (h-x)(k+1)xr_q s_d \alpha + xr_q s_q \\ =& kxr_d s_d + \frac{(x-1)x}{2} kr_q s_d \alpha + xr_q s_d \alpha + \frac{(h-x)(h-x+1)}{2}(k+1) r_d s_d \\ &+ (h-x)(k+1)xr_q s_d \alpha + xr_q s_q \end{aligned} \tag{3.1.7}$$

Vorremmo ora replicare quanto fatto nella Sezione 2.1.6 e cercare quindi di dimostrare la **bitonicità**, cioè che facendo scendere l'ultimo nodo nero x il costo totale prima decresce e poi decresce. In questo caso tuttavia esiste un controesempio che mostra come non valga la bitonicità, ed è presentato in Figura 3.1.5.

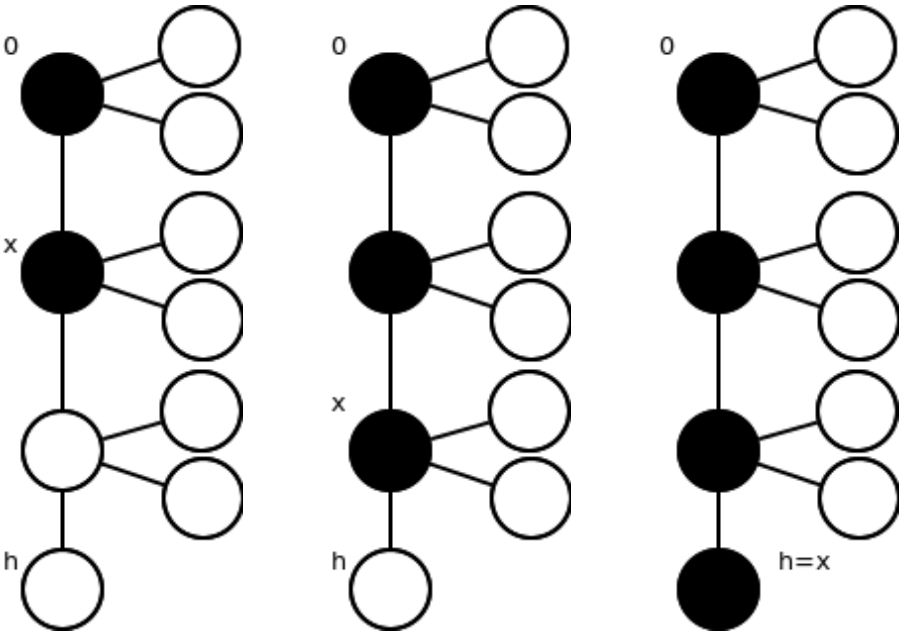
Non resta quindi che cercare esaustivamente il nodo x che minimizza $E(x)$.

3.1.5.1 Pseudocodice

Mostriamo ora lo pseudocodice; al termine della computazione la variabile E conterrà il costo della soluzione ottima.

Algoritmo 19: Modello 2, non limitato, caterpillar regolare

```
1  if r_d s_d > r_q s_d α + (e_re / (e_tr + e_re)) r_q s_q then
2  |  Usa l'Algoritmo 10;
3  else
4  |  E = +∞;
5  |  for x = 0 to h do
6  |  |  E(x) = kxr_d s_d + ((x-1)x/2) kr_q s_d α + xr_q s_d α +
7  |  |        + ((h-x)(h-x+1)/2)(k+1) r_d s_d + (h-x)(k+1)xr_q s_d α + xr_q s_q ;
8  |  |  E = min{E, E(x)};
9  |  end
10 end
```



(a) $E(1) = 11r_d s_d + 7r_q s_d \alpha + r_q s_q$ (b) $E(2) = 7r_d s_d + 11r_q s_d \alpha + 2r_q s_q$ (c) $E(3) = 18r_q s_d \alpha + 3r_q s_q$

Figura 3.1.5: Controesempio che mostra come non valga la bitonicità su caterpillar regolare nel caso $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$.
Per $k = 2, h = 3, r_d = s_d = r_q = 1, \alpha = \frac{99}{100}, e_{tr} = 1, e_{re} = 99, s_q = 0.05$ si dimostra che $E(1) < E(2) > E(3)$, e rimangono verificate le condizioni $\alpha r_q < r_d$ e $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$.
In particolare
$E(1) = 11 + 7\frac{99}{100} + 0.05 = 17.98$
$E(2) = 7 + 11\frac{99}{100} + 0.1 = 19.77$
$E(3) = 18\frac{99}{100} + 0.15 = 17.97$

L'Algoritmo 10 impiega tempo $O(1)$ per individuare la soluzione ottima e per ottenere il costo di tale soluzione. Il ciclo *For* è svolto $O(h)$ e ad ogni iterazione il calcolo di $E(x)$ richiede tempo costante. Il tempo per individuare la soluzione ottima ed il costo di tale soluzione è quindi $O(h)$. Per individuare il nodo x nel ramo *else* è sufficiente tenere traccia del nodo x che minimizza $E(x)$. Occorre poi tempo $O(k_u)$ per piazzare effettivamente in nodi neri nelle posizioni opportune.

3.2 Limitato

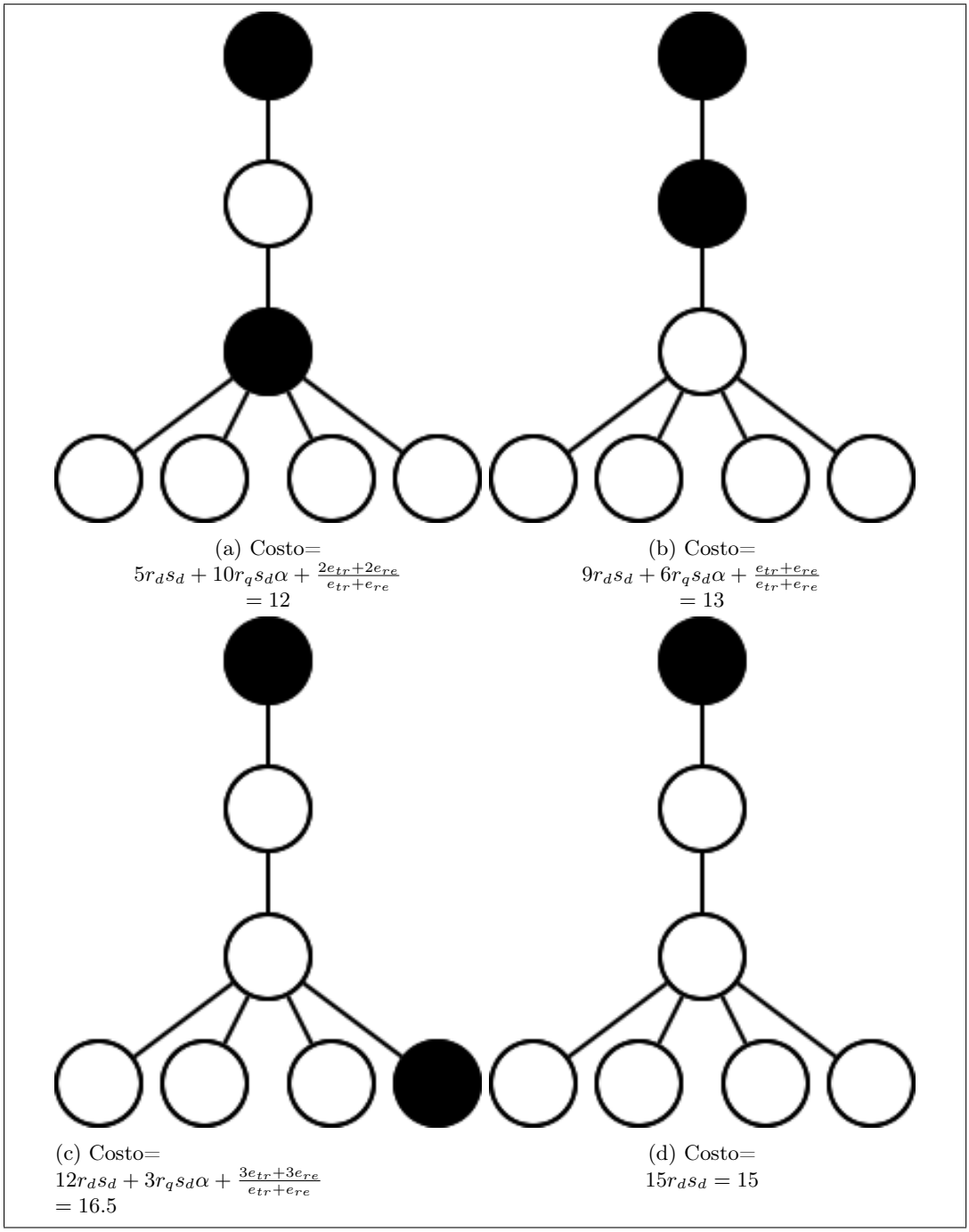
Passiamo ora ad analizzare i casi in cui abbiamo a disposizione un numero limitato $k \geq 1$ di nodi neri da piazzare. Come già visto nella Sezione 2.2 il problema del piazzamento è un tradeoff tra due costi contrapposti: i nodi neri devono rimanere vicini alla radice per diminuire il costo di distribuzione dell'interrogazione, ma dovrebbero anche evitare che i messaggi grezzi compiano un tragitto troppo lungo all'interno del'albero per diminuire il costo totale.

Similmente al caso limitato nel primo modello, non vale il Teorema 3.1.1 degli antenati in quanto esistono soluzioni ottime dove dei nodi neri hanno antenati bianchi, come ad esempio il caso mostrato in Figura 2.2.1 che può essere applicato al caso corrente con una lieve variazione, come mostrato in Figura 3.2.1.

Vale ancora il Teorema 1.1.3 secondo cui il conviene sempre mettere un nodo nero in un nodo che riceve già l'interrogazione (se abbiamo abbastanza neri a dispozione).

Studiando il caso limitato è sempre opportuno eseguire prima l'algoritmo corrispondente all'albero in input nel caso non limitato. Noi supporremo[4] che gli algoritmi per i casi non limitati abbiamo tempo di esecuzione inferiore, e ci indichino quanti nodi neri k_u usano. Se il numero di nodi neri a nostra disposizione è $k > k_u$ allora disporremo i neri come nel caso non limitato. *Per questi motivi gli algoritmi che presentiamo di seguito investigheranno il solo caso* $k < k_u$.

[4] Esamineremo tale supposizione nel capitolo 5.



(a) Costo= $5r_d s_d + 10r_q s_d \alpha + \frac{2e_{tr}+2e_{re}}{e_{tr}+e_{re}} = 12$

(b) Costo= $9r_d s_d + 6r_q s_d \alpha + \frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} = 13$

(c) Costo= $12r_d s_d + 3r_q s_d \alpha + \frac{3e_{tr}+3e_{re}}{e_{tr}+e_{re}} = 16.5$

(d) Costo= $15r_d s_d = 15$

Figura 3.2.1: Controesempio che mostra come nel caso limitato nel secondo modello un nodo nero possa avere antenati bianchi. Per $k = 2, r_d = s_d = r_q = s_q = e_{tr} = e_{re} = 1, \alpha = 0.5$ abbiamo verificato esaustivamente che la colorazione (a) è ottima. La colorazione (c) è identica a quelle ottenibili colorando una diversa foglia.

3.2.1 Albero

Studiamo il caso generico dove in input abbiamo un albero senza particolare topologia. L'algoritmo che presentiamo è una variante di quello presentato per il primo modello nella Sezione 2.2.1, da cui riprendiamo tutte le variabili e le definizioni.

Per tenere conto del fatto che l'interrogazione è inviata ai soli sottoalberi che hanno nodi neri, definiamo la quantità $Q_0^i(m,p)$ come il costo di invio dell'interrogazione da parte di i quando T_i contiene al più m neri, ed i sottoalberi di i si spartiscono i nodi neri secondo la permutazione p; ricordiano che se assegnamo ad un sottoalbero T_j al più $a > 0$ nodi neri, esso ne conterrà almeno uno per convenzione.

$$Q_0^i(m,p) = \begin{cases} 0 & \text{se } m = 0 \\ \frac{e_{tr}+B(p)e_{re}}{e_{tr}+e_{re}} r_q s_q & \text{se } m \geq 1 \end{cases}, Q_1^i(m,p) = Q_0^i(m-1,p) \tag{3.2.1}$$

dove $B(p)$ è il numero di elementi di p maggiori di zero. Tale quantità indica il numero di sottoalberi di i che contengono nodi neri e che quindi devono ricevere l'interrogazione. Il calcolo di $Q_0^i(m,p)$ richiede dunque tempo $O(c)$.

Possiamo ora ridefinire il valore $E_i(l,m)$ (mostrato nella Definizione 2.2.2 a pagina 34) come

Definizione 3.2.1.

Caso 1. se i è una foglia, $E_i(l,m)$ include il costo di i ed costo per l'invio fino alla radice di tutto l'albero.

$$E_i(m,l) = \begin{cases} (l+1)r_d s_d + (d_i - l)r_q \alpha s_d & \text{se } m = 0 \\ (d_i+1)r_q \alpha s_d & \text{se } m \geq 1 \end{cases} \tag{3.2.2}$$

Caso 2. se i è bianco non foglia, l'upper bound m deve essere diviso tra i $|C_i|$ figli. Sia $P(m)$ l'insieme di tutte le permutazioni $p = \left(m_j^p | j \in C_i\right)$, dove $\sum_{j \in C_i} m_j^p = m$ e m_j^p denota il massimo numero di neri nel sottoalbero T_j nella permutazione p. $E_i(l,m)$ è dunque definito dalla somma dei costi:

- $\min_{\forall p \in P(m)} \left\{ \sum_{j \in C_i} E_j\left(m_j^p, l+1\right) + Q_0^i(p,m) \right\}$: costo dei sottoalberi e costo di invio dell'interrogazione,
- $r_d s_d$: costo energetico di i senza invio dell'interrogazione,
- $l r_d s_d + (d_i - l) r_q \alpha s_d$: costo precalcolato per far giungere l'informazione di i alla radice.

Dunque sommando:

$$\begin{aligned} E_i(m,l) = \min_{\forall p \in P(m)} & \left\{ \sum_{j \in C_i} E_j\left(m_j^p, l+1\right) + Q_0^i(p,m) \right\} + \\ & + (l+1)r_d s_d + (d_i - l) r_q \alpha s_d \end{aligned} \tag{3.2.3}$$

Caso 3. se i è nero non foglia, l'upper bound $m-1$ deve essere diviso tra i figli C_i. Sia $P(m-1)$ l'insieme di tutte le permutazioni $p = \left(m_j^p | j \in C_i\right)$, dove $\sum_{j \in C_i} m_j^p = m-1$ e m_j^p denota il massimo numero di neri nel sottoalbero T_j nella permutazione p. Dunque $E_i(m,l)$ si calcola sommano i costi:

- $\min_{\forall p \in (P(m-1))} \left\{ \sum_{j \in C_i} E_j \left(m_j^p, 0 \right) + Q_1^i(p,m) \right\}$: costo dei sottoalberi e dell'invio dell'interrogazione,
- $r_q \alpha s_d$: il costo di i senza invio dell'interrogazione,
- $d_i r_q \alpha s_d$: costo precalcolato per far risalire l'informazione di i alla radice.

Sommando otteniamo:

$$E_i(m,l) = \min_{\forall p \in (P(m-1))} \left\{ \sum_{j \in C_i} E_j \left(m_j^p, 0 \right) + Q_1^i(p,m) \right\} + \\ + (d_i + 1) r_q \alpha s_d \tag{3.2.4}$$

Si noti che i valori $Q_0^i(p,m)$ e $Q_1^i(p,m)$ devono essere inclusi nel calcolo del minimo poiché dipendono da p.
Non è necessario cambiare altro nell'algoritmo.

3.2.1.1 Pseudocodice

Possiamo ora riscrivere l'Algoritmo 11 adattandolo al caso corrente.

Algoritmo 20: Modello 2, limitato, albero generico

```
1  foreach foglia i do
2      for m = 0 to k − 1 do
3          for l = 0 to n − 2 do
4              if m = 0 then
5                  E_i[m,l] = (l+1) r_d s_d + (d_i − l) r_q α s_d;
6              if m ≥ 1 then
7                  E_i[m,l] = (d_i + 1) r_q α s_d
               end
           end
       end
   end
8  foreach nodo i non foglia, non radice in postordine do
9      for m = 0 to k − 1 do
10         for l = 0 to n − 2 do
               // i bianco
11             min_1 = min_{∀p∈P(m)} {Σ_{j∈C_i} E_j[m_j^p, l+1] + Q_0^i(m,p)}+
                       +(l+1) r_d s_d + (d_i − l) r_q α s_d
               // i nero
12             min_2 = min_{∀p∈(P(m−1))} {Σ_{j∈C_i} E_j[m_j^p, 0] + Q_1^i(m,p)}+
                       +(d_i + 1) r_q α s_d
13             E_i[m,l] = min{min_1, min_2};
           end
       end
   end
14 E_n[k,0] = min_{∀p∈P(m−1)} { Σ_{j∈C_n} E_j [m_j^p, 0] + Q_1^i(m) } + r_q α s_d;
15 E_w = Σ_{i≠root} d_i r_d s_d;
16 E_n[k,0] = min{E_n[k,0], E_w};
```

Al termine della computazione $E_n[k,0]$ contiene il costo ottimo dell'albero. Il valore E_w rappresenta il costo dell'albero con soli nodi bianchi, tale costo deve essere calcolato a parte. Per conoscere la colorazione ottima occorre tenere traccia della colorazione di i durante il calcolo del minimo. Il calcolo di $Q_0^i(m,p)$ e $Q_1^i(m,p)$ richiede tempo $O(c)$, ma questo non varia la complessità dell'algoritmo che rimane $O(kn^2(\max\{k,c\})^{c-1})$.

3.2.2 Albero regolare completo

Supponiamo che l'albero in ingresso sia regolare e completo, cioè che ogni nodo non foglia abbia esattamente c figli e che tutte le foglie si trovino alla stessa profondità. Possiamo modificare l'Algoritmo 20 per alberi generici ottenendo una complessità minore, in modo del tutto simile a quanto visto nel primo modello su alberi regolari completi nella Sezione 2.2.2. L'algoritmo risultante è identico all'Algoritmo 12 con le osservazioni fatte nella Sezione 3.2.1.

Osserviamo che tutti i sottoalberi radicati alla stessa profondità sono alberi regolari completi con identica topologia. Ciò suggerisce di studiare il costo di ogni sottoalbero radicato a profondità diversa invece di ogni nodo. Numeriano i livelli in questo modo: le foglie sono tutte al livello 0, la radice avrà livello $\lfloor \log_c n \rfloor$, e tutti gli altri nodi saranno nominati di conseguenza. Ricordiamo la funzione di costo di un sottoalbero radicato in i, $E_i(l, m)$ esposta nella Definizione 3.2.1 e la adattiamo la caso corrente:

Definizione 3.2.2. sia $E_h(m, l)$ per $0 \leq m \leq k, 0 \leq l \leq \lfloor \log_c n \rfloor - 1$ il costo energetico del sottoalbero T_h con radice al livello h con al più m nodi neri, sommato al costo per far giungere le informazioni generate alla radice, dove l è il numero di nodi bianchi tra la radice del sottoalbero ed il più vicino antenato nero. Se $m > 0$ allora assumiamo che il sottoalbero contenga almeno un nodo nero.

Ridefiniamo ora il costo di invio dell'interrogazione da parte di un generico nodo quando il sottoalbero radicato in tale nodo contiene al più m nodi neri (almeno 1 se $m > 0$) ed i suoi sottoalberi si spartiscono gli m nodi neri secondo la permutazione p:

$$Q_0(m, p) = \begin{cases} 0 & \text{se } m = 0 \\ \frac{e_{tr} + B(p) e_{re}}{e_{tr} + e_{re}} r_q s_q & \text{se } m \geq 1 \end{cases}, Q_1(m, p) = Q_0(m - 1, p)$$

dove $B(p)$ è il numero di elementi di p maggiori di zero. Tale quantità indica il numero di sottoalberi di i che contengono nodi neri e che quindi devono ricevere l'interrogazione. Il calcolo di $Q_0(m, p)$ richiede dunque tempo $O(c)$. Poiché questa quantità dipende da p dovremo inserirla nel calcolo del minimo $E_h(m, l)$, come visto nell' Equazione 3.2.1 .

Sia inoltre $H = \lfloor \log_c n \rfloor$. L'algoritmo costruisce una tabella bi-dimensionale $E_h[m, l]$ per ogni livello h, contenente i valori $E_h(m, l)$, procede dal basso verso l'alto calcolando prima $E_h[m, l]$ per le foglie al livello 0, per $0 \leq m \leq k$ e $0 \leq l \leq H - 1$. Poi calcola $E_h[m, l]$ verso l'alto, dal livello 1 al livello $H - 1$. La radice è trattata separatamente poiché sarà sempre un nodo nero.

3.2.2.1 Pseudocodice

Mostriamo ora lo pseudocodice dell'algoritmo.

Algoritmo 21: Modello 2, limitato, albero regolare completo

```
1  H = ⌊log_c n⌋;
2  for m = 0 to k do
3  |  for l = 0 to H − 1 do
4  |  |  if m = 0 then
5  |  |  |  E_0[m, l] = (l + 1)r_d s_d + (H − l)r_q α s_d;
6  |  |  if m ≥ 1 then
7  |  |  |  E_0[m, l] = (H + 1)r_q α s_d
   |  |  end
   |  end
   end
8  for h = 1 to H − 1 do
9  |  for m = 0 to k do
10 |  |  for l = 0 to H − 1 do
   |  |  |  // i bianco
11 |  |  |  min_1 = min_{∀p∈P(m)}{Σ_{j=1}^{c} E_{h−1}[m_j^p, l + 1] + Q_0(m, p)}+
   |  |  |       +(l + 1)r_d s_d + (H − h − l)r_q α s_d;
   |  |  |  // i nero
12 |  |  |  min_2 = min_{∀p∈(P(m−1))}{Σ_{j=1} E_{h−1}[m_j^p, 0] + Q_1(m, p)}+
   |  |  |       +(H − h + 1)r_q α s_d;
13 |  |  |  E_h[m, l] = min{min_1, min_2};
   |  |  end
   |  end
   end
14 E_H[k, 0] = min_{∀p∈P(k−1)} { Σ_{j=1}^{c} E_{H−1} [m_j^p, 0] + Q_1(m, p) } + r_q α s_d;
```

L'algoritmo costruisce $\lfloor \log_c n \rfloor$ tabelle di dimensione $O(k) \times O(\log n)$. Come visto nell'Equazione 2.2.5 il costo per calcolare i minimi tenendo conto del numero di permutazioni $p(m)$ e del costo delle sommatorie al loro interno ha come upperbound $\max\{k, c\}^{c-1}$; Il calcolo di $Q_0(m, p)$ inoltre non aumenta tale costo. I cicli *For* vengono invece eseguiti $O(k(\log^2 n))$ volte. L'algoritmo presentato quindi ha complessità

$$O\left(k(\log^2 n)(\max\{k, c\})^{c-1}\right)$$

Per individuare la colorazione ottima è necessario tenere traccia del colore di i che minimizza $E_h[m, l]$ nel ciclo più interno, e del colore dei suoi sottoalberi.

Come gia visto nella Sezione 2.2.2 il numero di permutazioni può essere diminuito osservando che sono necessarie solo le diverse partizioni di m. Creando in anticipo tali partizioni possiamo descrivere un nuovo algoritmo con costo computazionale pari a

$$O\left(k(\log^2 n)\frac{\frac{\pi}{\sqrt{6}}e^{\left(\frac{2\pi}{\sqrt{6}}\sqrt{k}\right)}}{\sqrt{k}} + kc\right) \tag{3.2.5}$$

secondo quanto detto nella sezione di riferimento.

3.2.3 Cammino

Supponiamo ora di avere in input un cammino composto da n nodi dove ogni nodo ha esattamente un solo figlio, tranne il nodo n che sarà l'unica foglia.

Poiché ogni nodo ha un solo figlio si può dire che i nodi che inviano l'interrogazione la inviano a tutti i figli. Questa è la caratteristica tipica del primo modello di comunicazione, e suggerisce quindi che risolvere il problema limitato nel primo o nel secondo modello siano equivalente. Possiamo applicare quindi l'algoritmo descritto nella Sezione 2.2.3 a pagina 41 al caso corrente. Ricordiamo inoltre che tale algoritmo invididua l'ultimo nodo nero x^* della soluzione ottima in tempo $O(\log k)$ con una ricerca binaria. Occorre poi tempo $O(k)$ per individuare le posizioni e piazzare tutti i nodi neri, utilizzando l'Equazione 2.2.12.

3.2.4 Caterpillar

Si vuole ora risolvere il caso limitato quando l'albero di input è un caterpillar. Supponiamo che il nodo h sia sempre una foglia. Trasformeremo l'Algoritmo 14 addattandolo al caso corrente; richiameremo quindi le variabili e le definizioni presentare nella Sezione 2.2.4.

Sappiamo che il nero a maggior profondità è sicuramente nella dorsale, se così non fosse lo potremmo spostare nel fratello in dorsale senza che il costo totale aumenti. Si osservi inoltre che il Teorema 2.2.5 qui riportato

Teorema 3.2.3. *In un caterpillar se ignoriamo il costo di discesa dell'interrogazione, conviene sempre mettere un nero in dorsale piuttosto che in foglia.*

rimane valido anche nel secondo modello, quindi dopo aver individuato l'ultimo nodo nero x inizieremo ad annerire i nodi in dorsale in $Cat(o, x-1)$. Se $k \leq x+1$ allora i nodi neri saranno solo nella dorsale. Nel caso $k > x+1$ nel primo modello dopo aver annerito tutta la dorsale fino ad x avremmo annerito anche le foglie di $Cat(0, x-1)$ avendo sufficienti neri a disposizione; nel modello corrente invece questo dipende da alcuni parametri. Ricordiamo quanto detto nel caso non limitato su grafo caterpillar, in particolare il Teorema 3.1.7 a pagina 73

Teorema 3.2.4. *Se $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$ allora nella soluzione ottima tutte le foglie in $Cat(0, x-1)$ sono bianche, altrimenti sono tutte nere.*

è applicabile anche al caso limitato, con la medesima dimostrazione. Se dunque $k > x+1$ e $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$ allora porremo nere le foglie in $Cat(0, x-1)$ fino ad esaurire i nodi neri, avendo già annerita la dorsale. Facciamo osservare al lettore che non possiamo utilizzare l'Algoritmo 14 per risolvere questo caso, poiché può capitare che alcune foglie bianche abbiano fratelli neri: in questa occasione tali foglie riceverebbero l'interrogazione nel primo modello ma non la riceverebbero nel secondo, cambiando così il costo totale.

Sia $E(q, m)$ il costo ottimo del caterpillar $Cat(0, q)$ con al più m neri, che comprende la risalita delle informazioni fino alla radice e la ricezione dell'interrogazione da parte delle eventuali foglie nere; non comprende invece i costi di invio e ricezione dell'interrogazione da parte dei nodi nella dorsale; tale costo è definibile come:

$$E(q,m) = \begin{cases} B(q,m) & \text{se } m \geq q+1 \\ \min_{\forall x, m \leq x \leq q} \{E(x-1, m-1) + R(x,q)\} & \text{se } m < q+1 \end{cases} \tag{3.2.6}$$

ed $E(q,1) = R(0,q)$ dove

$R(i,j) =$ costo di risalita fino alla radice delle informazioni generate dal sottoalbero $Cat(i,j)$, sapendo che i è nero.

$B(q,m) =$ costo di $Cat(0,q)$ con al più m neri e $m \geq q+1$, e comprensivo del costo di ricezione dell'interrogazione da parte delle foglie nere. Non comprende i costi di invio e ricezione dell'interrogazione da parte dei nodi nella dorsale.

Nel primo caso ($m \geq q+1$) abbiamo neri a sufficienza per annerira tutta la dorsale ed eventualmente qualche foglia, e $B(q,m)$ calcola proprio questo caso; il caso $m = q+1$, cioè quando tutta la dorsale è nera ed abbiamo solo foglie bianche, rientra in questa parte dell'equazione. Nel secondo caso ($m < q+1$) tutte le foglie sono bianche e ci sarà almeno

un nodo bianco in dorsale tra i nodi $i, i \in [1, q]$; il valore $E(q, m)$ si calcola cercando esaustivamente l'ultimo nodo nero x e sommando poi il costo di $Cat(x, q)$ compreso di risalita fino alla radice, $R(x, q)$, e sommando il costo di $Cat(0, x-1)$ con un nodo nero in meno, che abbiamo già calcolato, $E(x-1, m-1)$. Si osservi che in questo caso sia $E(q, m)$ che $E(x-1, m-1)$ non hanno foglie nere. Poiché i nodi neri saranno solo in dorsale e ci deve essere almeno un bianco in dorsale, la ricerca di x è limitata a $m \leq x \leq q$, ricordiamo infatti che i nodi in dorsale sono indicizzati da 0, cioè la radice, fino ad h, l'ultimo nodo della dorsale (che è una foglia), e che nel numero di nodi neri m è compresa anche la radice nera. Se $m = 1$ allora solo la radice sarà nera è possiamo quindi dire $E(q, 1) = R(0, q)$.

Osservando il costo $E(q, m)$ si intuisce che quando rientriamo nel secondo caso, $m < q+1$, il costo è dato dalla somma dei costi di k caterpillar $Cat(a, b)$ che hanno la radice come unico nodo nero, e costo $R(a, b)$. Se dunque nella soluzione ottima non abbiamo foglie nere ed i nodi neri hanno indice $x_1, x_2, \ldots, x_k$, il costo totale (privo del costo di distribuzione dell'interrogazione) è calcolabile tramite

$$E = R(0, x_1 - 1) + R(x_1, x_2 - 1) + \ldots + R(x_{k-1}, x_k - 1) + R(x_k, h)$$

L'algoritmo utilizza la programmazione dinamica per calcolare i valori $E(q, m)$, $\forall q \in [0, h]$, $\forall m \in [2, k]$ per q ed m crescenti tramite l'Equazione 3.2.6. Il costo totale E dell'albero si trova poi cercando esaustivamente quale sia l'ultimo nodo in dorsale x che minimizza il costo totale. Dato x si può trovare E sommando il costo di invio e ricezione dell'interrogazione da parte dei nodi in dorsale fino ad x [5], $\frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} x r_q s_q = x r_q s_q$, sommando il costo di $Cat(0, x-1)$ con un nodo nero in meno, $E(x-1, k-1)$, e sommando il costo di $Cat(x, h)$ comprensivo della risalita fino alla radice, $R(x, h)$. Riassumendo:

$$E = \min_{\forall x, x \leq h} \{x r_q s_q + E(x-1, k-1) + R(x, h)\} \tag{3.2.7}$$

Possiamo calcolare $B(q, m)$ nel seguente modo:

$$B(q, m) = \begin{cases} F(0, q) r_d s_d + [M'_q - F(0, q)] r_q s_d \alpha & \text{se ..} \\ M'_q r_q s_d \alpha + F(0, q) \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q & \text{se ..} \\ (N'_q - m) \left(r_d s_d + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q \right) + [M'_q - (N'_q - m)] r_q s_d \alpha & \text{se ..} \end{cases} \tag{3.2.8}$$

Analizziamo meglio tale equazione ricordando che $B(q, m)$ è definito solo per $m \geq q + 1$, e quindi tutta la dorsale fino a x è nera:

- Il primo caso si applica se $r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$; in questa situazione le $F(0, q)$ foglie inviano un messaggio grezzo ciascuna, e gli altri $[M'_q - F(0, q)]$ messaggi sono di risposta.

- Il secondo caso si applica se $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q \wedge N'_q \leq m$, cioè quando le foglie possono essere nere ed il numero di nodi neri a disposizione è maggiore o uguale al numero di nodi in $Cat(0, q)$, ovvero $N'_q \leq m$. In questa situazione il caterpillar $Cat(0, q)$ sarà completamente nero, tutti i M'_q messaggi scambiati sono di risposta, e tutte le $F(0, q)$ foglie ricevono l'interrogazione.

- Il terzo caso si applicano quando $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q \wedge N'_q > m$. Quando ciò accade le foglie possono essere nere, ma non abbiamo sufficienti nodi neri a disposizione per annerirle tutte. Abbiamo quindi $(N'_q - m)$ foglie bianche che ricevono

[5] Il nodo 0 paga solo costo di invio ed il nodo x paga solo il costo di ricezione dell'interrogazione,

l'interrogazione ed inviano un messaggio grezzo ciascuna, e $[M'_q - (N'_q - m)]$ messaggi di risposta.

In precomputazione vengono calcolati i valori N'_q, N_q, M_q, M'_q, $\forall q \in [0, h]$; ricordando che $F(0, q) = N_q - q - 1$, allora i valori $B(q, m)$ sono calcolabili in tempo costante.

Possiamo poi calcolare i valori $R(i, j)$, sapendo che in $Cat(i, j)$ vengono scambiato un numero di messaggi pari a quelli scambiato in $Cat(i, h)$ meno quelli scambiati in $Cat(j+1, h)$ meno la risalita da $j+1$ ad i delle informazioni generate in $Cat(j+1, h)$, quindi $M_i - M_{j+1} - N_{j+1}(j - i + 1)$. A questo dobbiamo sommare la risalita di informazioni da i a 0 generate dai $N_i - N_{j+1}$ nodi in $Cat((i, j)$; riassumendo quindi

$$R(i, j) = [M_i - M_{j+1} - N_{j+1}(j - i + 1)]r_d s_d + [N_i - N_{j+1}]r_q s_d \alpha \tag{3.2.9}$$

Avendo precalcolato $N_q, M_q, \forall q \in [0, h]$, $R(i, j)$ è calcolabile in tempo costante. Avevamo già visto un esempio di calcolo di $R(i, j)$ in Figura 2.2.6 a pagina 53.

3.2.4.1 Pseudocodice

Possiamo ora esporre lo pseudocodice dell'algoritmo.

Algoritmo 22: Modello 2, limitato, caterpillar

```
1  Effettua le precomputazioni;
2  for q=0 to h do
3  |  E(q, 1) = R(0, q);
4  end
5  for m=2 to k-1 do
6  |  for q=0 to h-1 do
7  |  |  if m ≥ q + 1 then
8  |  |  |  E(q, m) = B(q, m);
9  |  |  else
10 |  |  |  E(q, m) = min_{∀x, m≤x≤q} {E(x − 1, m − 1) + R(x, q)};
11 |  |  end
12 |  end
13 end
14 E = min_{∀x, x≤h} {x r_q s_q + E(x − 1, k − 1) + R(x, h)};
```

Durante il calcolo di E si analizza il caso $x = 0$: definendo $E(q, m) = 0, \forall q < 0, m < 0$ allora viene calcolato il costo corretto $E = R(0, h)$. Le precomputazioni calcolano N_q, N'_q, M_q, M'_q tramite gli Algoritmi 5, 6, 7, 8 in tempo $O(h)$. Il primo ciclo ha complessità $O(h)$. Il secondo ciclo è composto da due cicli innestati ed un calcolo del minimo, che insieme danno il costo totale dell'algoritmo $O(h^2 k)$. Al termine della computazione la variabile E contiene il costo della soluzione ottima. Per ricavare la colorazione ottima è necessario tenere traccia dell'ultimo nodo nero x che minimizza ogni $E(q, m)$, e della colorazione di ogni $B(q, m)$. Sia β la colorazione ottima di $E(q, m)$.

Caso 1. quando $E(q, m) = B(q, m)$, β avrà tutta la dorsale nera, ed altre $m - q - 1$ foglie qualsiasi nere.

Caso 2. quando $E(q, m) = \min_{\forall x, m \le x \le q} \{E(x - 1, m - 1) + R(x, q)\}$, β avrà il nodo x nero, ed i suoi discendenti fino a q saranno bianchi. Gli altri nodi avranno la stessa colorazione di $E(x - 1, m - 1)$.

La colorazione ottima di tutto l'albero si ottiene durante il calcolo di E: in questo caso x sarà il nodo nero a profondità maggiore, ed i restanti nodi avranno la stessa colorazione di $E(x-1, k-1)$.

3.2.4.2 Dimostrazione di ottimalità e correttezza

Occorre dimostrare la corretta e l'ottimalità di $E(q,m)$, $B(q,m)$, $R(i,j)$ ed E. Noi assumiamo che $B(q,m)$, $R(i,j)$ ed E siano già ben definiti ed ottimi, in particolare che l'equazione 3.2.7 di E sia corretta. La seguente dimostrazione si limita quindi a provare che $E(q,m)$ è ben definito ed ottimo. Riscriviamo per comodità la definizione

$$E(q,m) = \begin{cases} B(q,m) & \text{se } m \geq q+1 \\ \min_{\forall x, m \leq x \leq q} \{E(x-1, m-1) + R(x,q)\} & \text{se } m < q+1 \end{cases}$$

ed ancora $E(q,1) = R(0,q)$.

Dimostrazione. Procediamo per induzione su m.
Per $m = 1$ possiamo avere solo la radice nera, dunque il costo di $E(q,1)$ coincide con la definizione di $R(0,q)$; questo caso è dunque corretto assumendo che sia ben definita $R(0,q)$.
Per $m > 1$ abbiamo, per ipotesi induttiva, che $E(q, m-1)$ è ben definito ed ottimo per qualsiasi q. Si distinguano altri due sottocasi:

Caso 1. $m \geq q+1$, dunque per il Teorema 2.2.5 tutta la dorsale sarà nera ed il suo costo si calcola con $B(q,m)$. Dunque questo caso è corretto se è ben definito $B(q,m)$.

Caso 2. $m < q+1$, allora $Cat(0,q)$ non ha foglie nere ed al suo interno cerchiamo esaustivamente l'ultimo nero x nella dorsale. Una volta individuato, il costo è dato da un caterpillar $Cat(x,q)$ in cui solo x è nero, dunque il suo costo si calcola con $R(x,q)$, e da un secondo caterpillar $Cat(0, x-1)$ in cui dobbiamo disporre ottimamente m-1 nodi neri, ed il cui costo è $E(x-1, m-1)$ che conosciamo già per ipotesi induttiva. Sempre in questo caso l'ottimalità è garantita dal fatto che la ricerca del nodo x è esaustiva. Possiamo anche mostrare la sottostruttura ottima della relazione di programmazione dinamica: $E(q,m)$ deve essere ottimo se i suoi sottocasi sono ottimi; infatti se fissato x, $E(x-1, m-1)$ non fosse ottimo, potremmo sostituire il suo valore ottimo $E'(x-1, m-1) < E(x-1, m-1)$ all'interno dell'equazione $E'(q,m) = E'(x-1, m-1) + R(x,q)$ ottenendo così un valore $E'(q,m) < E(q,m)$ inferiore all'ottimo che è ovviamente assurdo.

□

3.2.5 Caterpillar regolare

Supponiamo che l'albero in ingresso sia un caterpillar regolare dove ogni nodo ha esattamente c figli foglia; quindi tutti i nodi $[0, \ldots, h-1]$ avranno esattamente $c+1$ figli ed il nodo h ha c figli. L'algoritmo che presentiamo è una rivisitazione di quanto detto nel primo modello nella Sezione 2.2.5; la differenza principale consiste nel fatto che nel primo modello alcune foglie bianche possono ricevere l'interrogazione, mentre ciò non è possibile nel secondo modello di comunicazione.

L'algoritmo cerca l'ultimo nodo nero x in backbone e studia separatamente i casi $x+1 \geq k$ e $x+1 < k$.

3.2.5.1 Prima parte

Nel primo caso supponiamo $x+1 \geq k$, dunque tutti i nodi $[0, \ldots, x-1]$ ricevono l'interrogazione ed i nodi neri saranno piazzati solo in backbone, nelle posizioni $[0, \ldots, x]$ per il Teorema 2.2.5 (è meglio mettere i nodi neri in backbone piuttosto che nelle foglie); la dimostrazione del teorema rimane valida, a maggior ragione nel secondo modello di comunicazione, dove se ponessimo nera una foglia questa dovrebbe anche ricevere l'interrogazione, e ciò comporterebbe un ulteriore aumento di costo.

Sia a_i l'antenato nero più prossimo ad i, nel caso i sia nero allora $a_i = i$. Definiamo quindi

Definizione 3.2.5. Sia $C(i)$ il costo del nodo i e delle sue foglie, supponendo che tutte le sue foglie siano bianche. Questo comprende il costo di risalita fino alla radice e non comprende i costi di ricezione ed invio dell'interrogazione.

Possiamo calcolare $C(i)$ in funzione di a_i tramite la già presentata Equazione 2.2.22, qui riportata

$$C(i) = (i - a_i) r_d s_d (c+1) + c r_d s_d + a_i (c+1) r_q s_d \alpha \tag{3.2.10}$$

il costo energetico $E(x)$ dell'albero con ultimo nero x, si ricava sommando i costi :

- $C(i), \forall i \in [0, x]$: costo dei nodi in $Cat(0, x)$,
- $\sum_{i=x+1}^{h} [c r_d s_d + (c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha]$: costo di risalita fino alla radice delle informazioni generate in $Cat(x+1, h)$,
- $(x) r_q s_q \frac{e_{tr}+e_{re}}{e_{tr}+e_{re}} = x r q s_q$: diffusione dell'interrogazione.

Dunque sommando otteniamo:

$$\begin{aligned}
E(x) =& \sum_{i=0}^{x} C(i) + \sum_{i=x+1}^{h} [cr_d s_d + (c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha] \\
&+ x r_q s_q \\
=& \sum_{i=0}^{x} [(i-a_i) r_d s_d (c+1) + c r_d s_d + a_i (c+1) r_q s_d \alpha] \\
&+ \sum_{i=x+1}^{h} [cr_d s_d + (c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha] \\
&+ x r_q s_q \\
=& \sum_{i=0}^{h} c r_d s_d + (c+1) \sum_{i=0}^{x} [(i-a_i) r_d s_d + a_i r_q s_d \alpha] \\
&+ \sum_{i=x+1}^{h} [(c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha] \\
&+ x r_q s_q
\end{aligned} \tag{3.2.11}$$

a questo punto l'algoritmo potrebbe banalmente effettuare una ricerca del costo minimo tramite

$$E = \min_{x \in [k-1, h]} \{E(x)\}$$

vogliamo tuttavia applicare la tecnica studiata per il cammino nella Sezione 2.2.3 per cercare di individiare più velocemente quale sia x ottimo. Fissato x dobbiamo piazzare i nodi neri in modo da ottimizzare $E(x)$, e cioè variando le quantità a_i; in particolare ottimizzare $E(x)$ equivale ad ottimizzare solo

$$\begin{aligned}
E' =& \sum_{i=0}^{x} [(i-a_i)(r_d s_d) + a_i r_q s_d \alpha] = \\
=& \sum_{i=0}^{x} [(i-a_i)(r_q s_d \alpha + r_d s_d - r_q s_d \alpha) + a_i r_q s_d \alpha] = \\
=& \sum_{i=0}^{x} [i r_q s_d \alpha - a_i r_q s_d \alpha + (i-a_i)(r_d s_d - r_q s_d \alpha) + a_i r_q s_d \alpha] = \\
=& \sum_{i=0}^{x} (i r_q s_d \alpha) + \sum_{i=0}^{x} (i-a_i)(r_d s_d - r_q s_d \alpha)
\end{aligned} \tag{3.2.12}$$

che equivale ad ottimizzare solo il secondo termine. Nel caso corrente la colorazione dei nodi $[0, \ldots, x-1]$ è fatta da gruppi, ciascuno composto da un nodo nero seguito da nodi bianchi; riprendiamo quindi quanto scritto nell'Equazione 2.2.10, dove definendo $\sigma_j = i_{j+1} - i_j$ come la dimensione del j-esimo gruppo otteniamo

$$\begin{aligned}
C' =& \sum_{i=0}^{x} (i-a_i)(r_d s_d - \alpha r_q s_d) = \\
=& \frac{r_d s_d - \alpha r_q s_d}{2} \sum_{j=0}^{k-2} (\sigma_j^2 - x)
\end{aligned}$$

che sostituiamo in $E(x)$ all'interno dell'Equazione 3.2.11, ottenendo:

$$
\begin{aligned}
E(x) =& \sum_{i=0}^{h} c r_d s_d \\
&+ (c+1)\left[\sum_{i=0}^{x}(i r_q s_d \alpha) + \frac{r_d s_d - \alpha r_q s_d}{2}\sum_{j=0}^{k-2}(\sigma_j^2 - x)\right] \\
&+ \sum_{i=x+1}^{h}[(c+1)(i-x) r_d s_d + (c+1)(x) r_q s_d \alpha] \\
&+ x r_q s_q \\
=& \sum_{i=0}^{h} c r_d s_d \\
&+ (c+1)\left[\sum_{i=0}^{x}(i r_q s_d \alpha) + \frac{r_d s_d - \alpha r_q s_d}{2}\sum_{j=0}^{k-2}(\sigma_j^2 - x)\right] \\
&+ (c+1)\sum_{i=x+1}^{h}[(i-x) r_d s_d + (x) r_q s_d \alpha] \\
&+ x r_q s_q \\
=& (h+1) c r_d s_d + (c+1)\frac{x(x+1)}{2} r_q s_d \alpha \\
&+ (c+1)\frac{r_d s_d - r_q s_d \alpha}{2}\sum_{j=0}^{k-2}(\sigma_j^2 - x) \\
&+ (c+1)\sum_{i=1}^{h-x}[(i) r_d s_d + (x) r_q s_d \alpha] \\
&+ x r_q s_q \\
=& (h+1) c r_d s_d + (c+1)\frac{x(x+1)}{2} r_q s_d \alpha \\
&+ (c+1)\frac{r_d s_d - r_q s_d \alpha}{2}\sum_{j=0}^{k-2}(\sigma_j^2) \\
&- x(k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&+ (c+1)\frac{(h-x)(h-x+1)}{2} r_d s_d + (c+1)(h-x)(x) r_q s_d \alpha \\
&+ x r_q s_q \\
=& (h+1) c r_d s_d + (c+1)\frac{x(x+1)}{2} r_q s_d \alpha \\
&+ (c+1)\frac{r_d s_d - r_q s_d \alpha}{2}\sum_{j=0}^{k-2}(\sigma_j^2) \\
&- x(k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&+ (c+1)\frac{x(-2h-1) + x^2 + h^2 + h}{2} r_d s_d \\
&+ (c+1)(-x^2 + hx) r_q s_d \alpha \\
&+ x r_q s_q
\end{aligned}
\tag{3.2.13}
$$

Occorre dunque ottimizzare solo $\sum_{j=0}^{k-2} \sigma_j^2$, il cui minimo è dato dall'Equazione 2.2.12, poiché la sua dimostrazione è applicabile anche a questo caso. Quindi si ha il minimo con

$$\sigma_j^* = \begin{cases} \lceil \frac{x}{k-1} \rceil & 0 \leq j \leq |x|_{k-1} - 1 \\ \\ \lfloor \frac{x}{k-1} \rfloor & |x|_{k-1} \leq j \leq k-2 \end{cases} \tag{3.2.14}$$

da cui

$$\sum_{j=0}^{k-2} (\sigma_j^*)^2 = |x|_{k-1} \left\lceil \frac{x}{k-1} \right\rceil^2 + (k-1-|x|_{k-1}) \left\lfloor \frac{x}{k-1} \right\rfloor^2 \tag{3.2.15}$$

I gruppi hanno quindi la dimensione tra loro il più simile possibile: i primi hanno dimensione $\lceil \frac{x}{k-1} \rceil$ mentre quelli che seguono $\lfloor \frac{x}{k-1} \rfloor$. La disposizione relativa di questi gruppi non importa ai fini della minimizzazione di $\sum_{j=0}^{k-2} \sigma_j^2$, dunque abbiamo scelto arbitrariamente di disporli in questo modo. Possiamo dunque trovare $E(x)$ in tempo costante sostituendo l'Equazione 3.2.15 nell'Equazione 3.2.13.

Studiamo ora la funzione

$$\triangle E_{x+1} = E(x+1) - E(x)$$

per cercare un modo più efficre di individuare x ottimo. Spostando l'ultimo nero da x a $x+1$ il costo dell'albero aumenta se $\triangle E_{x+1} > 0$ e diminuisce se $\triangle E_{x+1} < 0$, inoltre aumentando x di un unità, uno dei gruppi di dimensione più piccola aumenta la sua dimensione di un unità. Il contributo dato a $\triangle E_{x+1}$ dalla somma dei quadrati è dunque

$$\left(2\left\lfloor \frac{x}{k-1} \right\rfloor + 1\right) \frac{r_d s_d - \alpha r_q s_d}{2} (c+1) > 0$$

come già mostrato nell'Equazione 2.2.15 a pagina 45.

Riscriviamo $\triangle E_{x+1}$ come

$$
\begin{aligned}
&=\triangle E_{x+1} = E(x+1) - E(x) = \\
&=(h+1)cr_d s_d - (h+1)cr_d s_d \\
&\quad + (c+1)\frac{(x+1)(x+2)}{2} r_q s_d \alpha - (c+1)\frac{x(x+1)}{2} r_q s_d \alpha \\
&\quad + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \sum_{j=0}^{k-2} (\sigma_j'^2 - \sigma_j^2) \\
&\quad - (x+1)(k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} + x(k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&\quad + (c+1)\frac{(x+1)(-2h-1) + (x+1)^2 + h^2 + h}{2} r_d s_d \\
&\qquad - (c+1)\frac{x(-2h-1) + x^2 + h^2 + h}{2} r_d s_d \\
&\quad + (c+1)(-(x+1)^2 + h(x+1)) r_q s_d \alpha - (c+1)(-x^2 + hx) r_q s_d \alpha \\
&\quad + (x+1) r_q s_q - (x) r_q s_q = \\
&=(c+1)\frac{x+1}{2} r_q s_d \alpha (x + 2 - x) \\
&\quad + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \left(2 \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) \\
&\quad - (k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&\quad + (c+1)\frac{(-2h-1) + 2x + 1}{2} r_d s_d \qquad (3.2.16) \\
&\quad + (c+1)(-2x - 1 + h) r_q s_d \alpha \\
&\quad + r_q s_q \\
&=(c+1)(x+1) r_q s_d \alpha + (c+1) x r_d s_d + (c+1)(-2x) r_q s_d \alpha \\
&\quad + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \left(2 \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) \\
&\quad - (k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&\quad + (c+1)\frac{(-2h-1) + 1}{2} r_d s_d \\
&\quad + (c+1)(-1 + h) r_q s_d \alpha \\
&\quad + r_q s_q \\
&=x(c+1)(r_d s_d - r_q s_d \alpha) + (c+1) r_q s_d \alpha \\
&\quad + (c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \left(2 \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) \\
&\quad - (k-1)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} \\
&\quad + (c+1)\frac{(-2h-1) + 1}{2} r_d s_d \\
&\quad + (c+1)(-1 + h) r_q s_d \alpha \\
&\quad + r_q s_q
\end{aligned}
$$

che coincide con l'Equazione 2.2.28 a pagina 59, con il costo di distribuzione dell'interrogazione corretto. Si evince che $\triangle E_{x+1}$ è strettamente crescente in x, dunque $E(x)$ ha un comportamento bitonico prima decrescente e poi crescente ed avrà il minimo in corrispondenza del valore x^* che trasforma $\triangle E_{x+1}$ da negativa a positiva. Si può dunque trovare $x^* \in [k-1, h]$ che mininizza $E(x)$ in tempo $O(\log h)$ con una ricerca bitonica; esiste tuttavia un metodo migliore: sostituiamo $x = m(k-1)$ con m intero $1 \leq m \leq \left\lfloor \frac{n-1}{k-1} \right\rfloor$ all'interno dell'Equazione 3.2.16 ottenendo che

$$m = \left\lceil -\frac{(2-k)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} + h(c+1)(r_q s_d \alpha - r_d s_d) + r_q s_q}{k(r_d s_d - r_q s_d \alpha)} \right\rceil \tag{3.2.17}$$

è il primo m per cui l'Equazione 3.2.16 diventa positiva. É dunque possibile restringere la ricerca di x^* a $(m-1)(k-1) \leq x^* \leq m(k-1)$ utilizzando così solo tempo $O(\log k)$ per la ricerca binaria. Si noti che questa sequenza può contenere al massimo h elementi, dunque nel caso $h < k$ la ricerca binaria impiegherà tempo $O(\log h)$.

3.2.5.2 Seconda parte

Il secondo caso che dobbiamo studiare è $x+1 < k$. Fissato x sappiamo che tutta la backbone $[0, x]$ deve essere nera; non è tuttavia sicuro che anche le foglie siano nere. Possiamo applicare il Teorema 3.1.7 a pagina 73 anche al caso corrente [6], dunque anneriremo le foglie di $Cat(0, x-1)$ se $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$ e se abbiamo sufficienti nodi neri a disposizione. Ricordiamo che nel caso corrente vale $x + 1 < k$, dunque avremo almeno una foglia nera.

Sia $E'(x)$ il costo dell'albero quando $x + 1 < k$ ed x è l'ultimo nodo nero. Tale costo si calcola sommando i costi:

- $\sum_{i=x}^{h} [c r_d s_d + r_d s_d (1+c)(i-x)]$: costo di risalita fino a x delle informazioni generate dai nodi in $Cat(x, h)$,
- $x \sum_{i=x}^{h} (c+1) \alpha r_q s_d$: risalita di questi stessi dati da x fino alla radice,
- $\sum_{i=1}^{x-1} i r_q s_d \alpha$: risalita fino alla radice dei dati generati dai nodi $[0, \ldots, x-1]$,
- $\sum_{i=1}^{x-1} i c r_q s_d \alpha$: costo di risalita delle informazioni generate dalle foglie di $Cat(0, x-1)$ contando la sola porzione di percorso dal padre delle foglie fino alla radice,
- $x r_q s_q$: costo di invio e ricezione dell'interrogazione fino ad x,
- $A(x)$: costo di invio delle informazioni dalle foglie in $Cat(0, x-1)$ al loro padre ed il costo di ricezione dell'interrogazione da parte della foglie nere.

Riscriviamo dunque $E'(x)$ come

[6] Il teorema chiama q la variabile qui denominata x.

$$
\begin{aligned}
E'(x) =& \sum_{i=x}^{h}[cr_d s_d + r_d s_d(1+c)(i-x)] + x\sum_{i=x}^{h}(c+1)\alpha r_q s_d \\
&+ \sum_{i=1}^{x-1} i r_q s_d \alpha + \sum_{i=1}^{x-1} i c r_q s_d \alpha \\
&+ x r_q s_q \frac{e_{tr} + (c+1)e_{re}}{e_{tr}+e_{re}} + A(x) \\
=& \sum_{i=0}^{h-x} cr_d s_d + \sum_{i=0}^{h-x}[r_d s_d(1+c)i] + x\sum_{i=0}^{h-x}(c+1)\alpha r_q s_d \\
&+ \sum_{i=1}^{x-1} i(c+1) r_q s_d \alpha \\
&+ x r_q s_q \frac{e_{tr} + (c+1)e_{re}}{e_{tr}+e_{re}} + A(x) \\
=&(h-x+1)cr_d s_d + \frac{(h-x)(h-x+1)}{2} r_d s_d(1+c) + x(h-x+1)(c+1)\alpha r_q s_d \\
&+ \frac{x(x-1)}{2}(c+1) r_q s_d \alpha + x r_q s_q \frac{e_{tr} + (c+1)e_{re}}{e_{tr}+e_{re}} + A(x)
\end{aligned}
\tag{3.2.18}
$$

Tra le cx foglie di $Cat(0, x-1)$ esattamente $k-x-1$ saranno nere mentre le restanti $cx-(k-x-i)$ rimarranno bianche se $r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q$; in caso contrario tutte le foglie saranno bianche. Le foglie sono tra loro indistinguibili, non importa dunque quali renderemo effettivamente nere. Osservando che il numero di foglie cx potrebbe essere minore dei nodi neri a disposizione, possiamo definire $A(x)$ in questo modo:

$$
A(x) = \begin{cases}
cx r_d s_d & \text{se } r_d s_d \leq r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q \\
cx[\alpha r_q s_d + \frac{e_{re}}{e_{tr}+e_{re}}] & \text{se } r_d s_d > r_q s_d \alpha + \frac{e_{re}}{e_{tr}+e_{re}} r_q s_q \wedge cx \leq k-x-1 \\
(k-x-1)[\alpha r_q s_d + \frac{e_{re}}{e_{tr}+e_{re}}] & +[cx-(k-x-1)] r_d s_d \qquad \text{altrimenti}
\end{cases}
\tag{3.2.19}
$$

Nel primo caso abbiamo solo foglie bianche per ipotesi, nel secondo abbiamo solo foglie nere e nel terzo abbiamo esaurito i $(k-x-1)$ neri rimasti, così le restanti foglie sono bianche. Sostituendo $A(x)$ all'interno dell'Equazione 3.2.18 possiamo calcolare $E'(x)$ in tempo $O(1)$.

3.2.5.3 Pseudocodice

Mostriamo ora lo pseudocodice dell'algoritmo:

Algoritmo 23: Modello 2, limitato, caterpillar regolare

$E = +\infty$;
// Caso $x + 1 < k$
for $x = 0$ *to* $k - 1$ **do**
  $E = \min\{E, E'(x)\}$;
end
$m = \left\lceil -\frac{(2-k)(c+1)\frac{r_d s_d - r_q s_d \alpha}{2} + h(c+1)(r_q s_d \alpha - r_d s_d) + r_q s_q}{k(r_d s_d - r_q s_d \alpha)} \right\rceil$;
// Caso $x + 1 \geq k$
foreach x *where* $(m-1)(k-1) \leq x \leq m(k-1)$, *con ricerca binaria* **do**
  $E = \min\{E, E(x)\}$;
end

Al termine della computazione E contiene il costo ottimo dell'albero. Il primo ciclo impiega tempo $O(k)$ mentre la ricerca binaria effettua $O(\log k)$ cicli ognuno a tempo costante. Nel caso $h < k$, il primo ciclo effettuerà al più h iterazioni, e la ricerca binaria analizzerà $O(\log h)$ elementi. Il costo dell'algoritmo è quindi $O(\min\{h, k\})$. Per piazzare i nodi neri al termine della computazione è necessario sempre tempo $O(k)$.

Per ricostruire la colorazione ottima occorre tenere traccia di quale x^* minimizza E: se x^* viene trovato nel primo ciclo, allora occorre porre neri tutti i nodi $[0, x^*]$ e porre nere tutte le foglie in $Cat(0, x^* - 1)$ fino ad esaurire i nodi neri. Se invece x^* viene individuato nel secondo ciclo, la posizione nei nodi neri si ottiene calcolando σ_j^* tramite l'Equazione 2.2.26 sostituendo x con x^*. Questo algoritmo è identico all'Algoritmo 15 per quanto riguarda lo pseudocodice e le tecniche utilizzate, ma i due hanno differenti definizioni di $E(x)$ ed $E'(x)$.

Capitolo 4

Modello 3

Vogliamo ora cambiare drasticamente il modello di comunicazione rispetto a quelli analizzati in precedenza. Nel terzo modello che analizziamo ci poniamo nel caso in cui le comunicazioni non avvengano tramite invio in broadcast, ma solo con invio punto a punto di informazioni. Il mittente deve quindi spendere una quantità e_{tr} di energia per ogni figlio a cui vuole inviare l'interrogazione. Restano invariate le altre proprietà: il costo energetico è ancora proporzionale solo alla dimensione del pacchetto dati scambiato, e non dipende dalla distanza fisica tra i nodi. Si noti che questo modello può essere visto come cablato con connessioni punto a punto, dove l'albero in input definisce la topologia della rete.

Il mittente ed il destinatario dovranno pagare un costo energetico per poter scambiarsi informazioni. Siano sempre e_{tr} ed e_{re} rispettivamente i costi energetici di invio e ricezione di un unità di informazione. L'invio verso il padre di informazioni rimane invariato rispetto ai precedenti modelli, mentre la distribuzione dell'interrogazione cambia sensibilmente: ora si rende necessario effettuare una trasmissione verso ogni figlio destinatario della trasmissione. Possiamo riscrivere il costo di invio non normalizzato dato dall'Equazione 1.1.1 a pagina 4 (dove il nodo i invia un unità di informazione vero il nodo j) come

$$\begin{cases} e_{tr} + e_{re} & \text{se } j \text{ è padre di } i \\ v_i(e_{tr} + e_{re}) & \text{altrimenti} \end{cases}$$

dove v_i è il numero di destinatari a cui i vuole inviare il messaggio.

Normalizzando tale costo per $(e_{tr} + e_{re})$ otteniamo

$$\begin{cases} 1 & \text{se } j \text{ è padre di } i \\ v_i & \text{altrimenti} \end{cases} \tag{4.0.1}$$

possiamo quindi risolvere ogni caso con algoritmi appositi ignorando i costi di invio e ricezione salvo poi rimoltiplicare l'energia totale E per $e_{tr} + e_{re}$ ottenendo così il vero costo energetico totale.

Nel secondo modello, il costo energetico per l'invio era definito dall'Equazione 3.0.1 a pagina 63 come

$$\begin{cases} 1 & \text{se } j \text{ è padre di } i \\ b_i = \frac{e_{tr}+e_{re}r}{e_{tr}+e_{re}} & \text{altrimenti} \end{cases} \tag{4.0.2}$$

dove r è il numero di destinatari dell'interrogazione. Ricordiamo che nella Sezione 1.1 abbiamo definito i vincoli $e_{tr} \geq 0, e_{re} \geq 0, e_{tr} + e_{re} > 0$. Possiamo quindi porre

$$e'_{tr} = 0, e'_{re} = 1$$

ed applicare tali parametri agli algoritmi definiti per il secondo modello, ottendendo un costo energetico

$$\begin{cases} 1 & \text{se } j \text{ è padre di } i \\ b_i = r & \text{altrimenti} \end{cases} \tag{4.0.3}$$

che coincide con quello definito per il terzo modello nell'Equazione 4.0.1.

Questo di dice che possiamo risolvere il problema definito sul terzo modello, con gli algoritmi definiti per il secondo modello, utilizzando gli algoritmi specifici per l'albero che abbiamo in input.

Presentiamo di seguito alcuni algoritmi specifici per il Modello 3. Si noti che vale ancora il Corollario 2.0.6 a pagina 8 (se $\alpha r_q \geq r_d$ allora ogni nodo sarà bianco), *quindi tutti gli algoritmi che presenteremo assumeranno di essere nel caso* $\alpha r_q < r_d$. Ricordiamo che anche nel modello corrente non è vero il contrario, cioè la soluzione ottima potrebbe avere solo nodi neri (con radice sempre nera), ma essere nel caso $\alpha r_q < r_d$.

Per semplicità di trattazione, assumeremo che il costo di invio dell'interrogazione $\frac{e_{tr}}{e_{tr}+e_{re}}$ sia sempre assegnato al nodo ricevente.

4.1 Non limitato

Studiamo prima il caso in cui abbiamo a disposizione un qualsiasi numero di nodi neri. Nel modello 3, come nei precedenti, vale ovviamente il Teorema degli antenati:

Teorema 4.1.1. ***Teorema degli antenati***
Nella colorazione ottima di un albero, se i è un nodo nero, allora esso avrà tutti gli antenati neri.

Da questo si evince che un nodo bianco avrà solo discendenti bianchi. Ciò è banalmente vero poiché tutti gli antenati di un nodo nero ricevono l'interrogazione e quindi abbiamo un risparmio energetico ponendoli neri.

Vale ovviamente anche il Teorema della frontiera:

Teorema 4.1.2. ***Teorema della frontiera***
La soluzione ottima nel caso non limitato è data da un insieme di nodi neri detto ***Frontiera****, che hanno solo antenati neri e almeno un figlio bianco (o nesssun figlio).*

Si noti che a differenza del Modello 1, qui i nodi della frontiera possono avere anche figli neri.

4.1.1 Albero

Esaminiamo il caso in cui l'albero in ingresso non abbia nessuna particolare topologia. L'algoritmo che presentiamo non ha un costo computazionale migliore dell'Algoritmo 17 per il Modello 2, ma utilizza una tecnica di risoluzione diversa dalla programmazione dinamica.

Definiamo il costo $e(i)$ di un nodo i in modo simile a quanto presentato nella Definizione 1.1.1 a pagina 4 assegnando però il costo di invio dell'interrogazione al nodo ricevente (il costo energetico normalizzato che usiamo è definito dall'Equazione 4.0.1). Questo costo comprende l'invio di tutti i messaggi grezzi e di risposta che transitano dal nodo i e la ricezione dell'interrogazione da parte di i.

Definizione 4.1.3.

Caso 1. $e(i) = |T_i| r_d s_d$, se i è bianco; tutti i dati generati dai $|T_i|$ nodi in T_i devono essere inviati al padre.

Caso 2. $e(i) = |T_i| r_q \alpha s_d + r_q s_q$, se i è nero e non ha discendenti neri; tutti i dati generati dai sensori in T_i sono restituiti come messaggi di risposta ed i riceve l'interrogazione.

Caso 3. $e(i) = |T_i| r_q \alpha s_d + r_q s_q$, se i è nero ed ha dei discendenti neri; i dati generati dai sensori in T_i sono restituiti da i in risposta all'interrogazione ed i riceve l'interrogazione.

Possiamo poi definire il costo totale di un albero T come

$$E(T) = \sum_{i \in T}^{e(i)}$$

Cerchiamo ora una condizione per cui un nodo possa essere nero o bianco. Siano T' e T'' due alberi con la stessa topologia e la stessa colorazione (con una frontiera che separa i nodi neri dai bianchi) a meno del nodo i. In particolare T'_i contiene solo nodi bianchi mentre T''_i ha radice nera e tutti gli altri nodi bianchi (si noti che il padre di i è sempre nero). La differenza tra i costi dei due alberi sarà pari a

$$E(T'') - E(T') = |T_i| r_d s_d - (r_q s_q + |T_i| r_q s_d \alpha)$$

dove $|T_i|$ è il numero di nodi contenuti nel sottoalbero T_i. I due alberi infatti si differenziano solo per il costo del nodo i. Quando questa quantità è maggiore di zero

$$\begin{aligned} &|T_i|(r_d s_d - r_q s_d \alpha) - r_q s_q > 0 \\ &|T_i| > \frac{r_q s_q}{r_d s_d - r_q s_d \alpha} \end{aligned} \tag{4.1.1}$$

conviene porre nero il nodo i per minimizzare il costo totale. In caso contrario conviene porre i bianco. Si noti che la condizione dipende dal numero di nodi nel sottoalbero T_i, da questo ricaviamo i seguenti Lemmi:

Lemma 4.1.4. *Se la condizione $|T_i| > \frac{r_q s_q}{r_d s_d - r_q s_d \alpha}$ è valida per il sottoalbero T_i radicato in i, allora è valida anche per tutti i sottoalberi radicati negli antenati di i.*

Lemma 4.1.5. *Se la condizione $|T_i| > \frac{r_q s_q}{r_d s_d - r_q s_d \alpha}$ non è valida per il sottoalbero T_i radicato in i, allora non è valida anche per tutti i sottoalberi radicati discendenti di i.*

Questi sono banalmente veri poiché se j è antenato di i allora $|T_j| > |T_i|$.

Dimostriamo ora il seguente Teorema:

Teorema 4.1.6. *Nella soluzione ottima saranno neri tutti i nodi i tali per cui vale*

$$|T_i| > \frac{r_q s_q}{r_d s_d - r_q s_d \alpha}$$

e saranno bianchi tutti i nodi j per cui essa non vale.

Poiché per gli antenati di i la condizione 4.1.1 rimane valida, questo è consistente con il Teorema 4.1.1 degli Antenati e con il Teorema 4.1.2 della Frontiera.

Dimostrazione. Supponiamo per assurdo che applicando il Teorema predecente, ci venga restituita una colorazione A dell'albero non ottima, e che la colorazione ottima dell'albero sia invece B; entrambe le colorazioni devono rispettare il Teorema 4.1.2 della Frontiera[1], avranno quindi due diverse frontiere. Analizziamo i due diversi casi che possono presentarsi, mostrati anche nella Figura 4.1.1

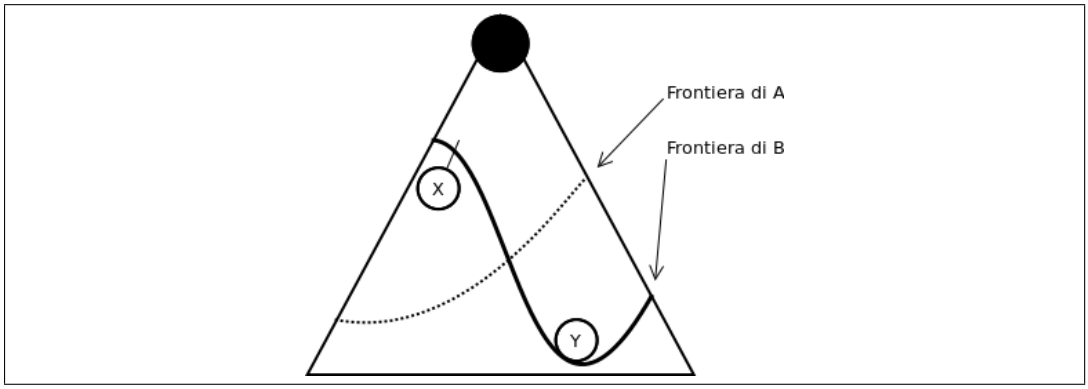


Figura 4.1.1: Qui è rappresentato l'albero, con la frontiera di A individuata dall'algoritmo, e la frontiera di B supposta ottima per assurdo. Si notino il nodo x figlio della frontiera di A ed il nodo y appartenente alla frontiera di B.

1. Esistono dei nodi che sono neri in A e bianchi in B; tra questi ve ne sarà in particolare uno, detto x, figlio di un nodo della frontiera di B. Essendo x nero in A per l'albero in esso radicato vale la condizione 4.1.1, ma allora possiamo porre x nero in B ed ottenere una nuova colorazione con costo inferiore a quella ottima, e ciò è ovviamente assurdo[2].

2. Esistono dei nodi bianchi in A e neri in B; tra questi selezioniamo un nodo y che appartenga alla frontiera di B. Poiché y è bianco in A, per l'albero in esso radicato non vale la condizione 4.1.1, ma allora possiamo porre y bianco in B ed ottenere un costo inferiore a quello ottimo, e ciò è assurdo.

□

[1] Se B non rispettasse il Teorema della Frontiera, cioè la sua soluzione non fosse data da una frontiera che separa i nodi neri da quelli bianchi, allora sarebbe non ottima banalmente.

[2] Essendo x figlio della frontiera, non ci saranno altri costi aggiuntivi per far giungere l'interrogazione ad x.

4.1.1.1 Pseudocodice

Mostriamo ora lo pseudocodice dell'algoritmo. L'algoritmo procede ponendo neri tutti i nodi per cui è verificata la condizione 4.1.1.

Algoritmo 24: Modello 3, non limitato, albero generico

```
1 Precomputazioni;
2 foreach nodo i do
3 |   if |T_i| > r_q s_q / (r_d s_d − r_q s_d α) then
4 |   |   Poni i nero;
5 |   end
6 end
```

L'algoritmo è corretto ed ottimo, infatti il Teorema 4.1.6 ci garantische che la condizione $|T_i| > \frac{r_q s_q}{r_d s_d - r_q s_d \alpha}$ vale per tutti i nodi appartenenti alla frontiera ottima, ed il Lemma 4.1.4 garantisce che essa vale per tutti i nodi antenati della frontiera.

L'algoritmo calcola durante la precomputazione $|T_i|, \forall i$ in tempo $O(n)$. Il tempo totale utilizzato dall'algoritmo è $O(n)$. Possiamo inoltre ricavare il costo E della soluzione ottima in tempo $O(n)$ con la formula $E = \sum_{i \in T}^{e(i)}$ utilizzando la Definizione 4.1.3.

4.1.2 Albero regolare completo

Studiamo ora il caso in cui l'albero in ingresso sia regolare e completo, cioè ogni nodo ha esattamente c figli e tutte le foglie sono alla stessa profondità. L'algoritmo qui presentato ha un costo inferiore a quello mostrato per il Modello 2.

Ricordiamo che h è la profondità delle foglie e che i sottoalberi radicati alla stessa profondità sono anch'essi alberi regolari completi ed hanno la stessa topologia.

Si consideri il costo $e(i)$ di un nodo i come esposto nella definizione 4.1.3. Partendo da questa, definiamo il costo di un sottoalbero T_l dove l è la profondità della sua radice, come la somma dei costi che incorrono in ogni nodo:

$$E(l) = E(T_l) = \sum_{i \in T_l} e(i)$$

Sia $|T_l|$ il numero di nodi di un sottoalbero radicato alla profondità l; esso si può calcolare ricordando l'Equazione 2.1.5 a pagina 16, dalla quale deriva:

$$|T_l| = \frac{c^{h-l+1} - 1}{c - 1}$$

Dalla Sezione 4.1.1 ricordiamo che se è valida la condizione esposta nell'Equazione 4.1.1

$$|T_i| > \frac{r_q s_q}{r_d s_d - r_q s_d \alpha}$$

allora conviene porre il nodo i nero; viceversa se non vale tale condizione, conviene porre i bianco (essendo nel caso in cui il nodo padre di i è nero e tutti i discendenti di i sono bianchi). Poiché data la profondità di un nodo conosciamo anche la sua topologia, possiamo riscrivere tale condizione come:

$$|T_l| > \frac{r_q s_q}{r_d s_d - r_q s_d \alpha} \tag{4.1.2}$$

Possiamo ora enunciare il seguente teorema:

Teorema 4.1.7. *L'Equazione* $|T_l| > \frac{r_q s_q}{r_d s_d - r_q s_d \alpha}$ *vale per i nodi della frontiera e non vale per i loro figli.*

Dimostrazione. Supponiamo di conoscere la soluzione ottima.

- Sia i un nodo nero della frontiera con profondità $d_i = l$; deve dunque valere $|T_l|(r_d s_d - r_q s_d \alpha) - r_q s_q > 0$ altrimenti potremmo sostituire i con un nodo bianco, diminuendo così il costo del sottoalbero radicato in i e conseguentemente diminuire il costo totale dell'albero, ma ciò è assurdo poiché abbiamo supposto che i appartenesse alla frontiera nella soluzione ottima.

- Sia i un nodo figlio di un nodo della frontiera; per i deve valere $|T_l|(r_d s_d - r_q s_d \alpha) - r_q s_q \leq 0$ altrimenti potremmo porre i nero, facendo diminuire di conseguenza il costo del sottoalbero T_i, e quindi trovando un costo totale dell'albero minore di quello di partenza, ma ciò è assurdo poiché eravamo già nel caso ottimo.

□

Si noti che se la condizione predecente vale per i sottoalberi radicati a profondità l, allora vale banalmente per tutti gli alberi radicati a profondità $f, f \leq l$, poiché quest contengono un numero maggiore di nodi.

Possiamo riscrivere l'Equazione 4.1.2 come

$$\begin{aligned}
|T_l| &> \frac{r_q s_q}{r_d s_d - r_q s_d \alpha} \\
\frac{c^{h-l+1} - 1}{c-1} &> \frac{r_q s_q}{r_d s_d - r_q s_d \alpha} \\
c^{h-l+1} &> \frac{r_q s_q (c-1)}{r_d s_d - r_q s_d \alpha}(c-1) + 1 \\
\log_c c^{h-l+1} &> \log_c \frac{r_q s_q (c-1)}{r_d s_d - r_q s_d \alpha}(c-1) + 1 \\
h - l + 1 &> \log_c \left(\frac{r_q s_q (c-1)}{r_d s_d - r_q s_d \alpha}(c-1) + 1 \right) \\
l &\leq -\log_c \left(\frac{r_q s_q (c-1)}{r_d s_d - r_q s_d \alpha}(c-1) + 1 \right) + h + 1
\end{aligned}$$

Per tutti gli alberi radicati a livelli l tali per cui è soddisfatta l'equazione precedente, vale la condizione esposta nell'Equazione 4.1.2, dunque preferiranno avere radice nera piuttosto che radice bianca (con discendenti bianchi). In particolare ciò è vero per tutti gli alberi la cui profondità l è $l \leq l'$

$$l' = \left\lfloor -\log_c \left(\frac{r_q s_q (c-1)}{r_d s_d - r_q s_d \alpha}(c-1) + 1 \right) + h + 1 \right\rfloor \tag{4.1.3}$$

Dunque per rispettare il Teorema 4.1.7 l'unica possibilità è che la frontiera sia composta dai soli nodi a profondità l'. Da qui si evince anche che i nodi della frontiera sono tutti alla stessa profondità.

4.1.2.1 Pseudocodice

Mostriamo per completezza lo pseudocodice.

Algoritmo 25: Modello 3, non limitato, albero regolare completo

1 $l' = \left\lfloor -\log_c \left(\frac{r_q s_q (c-1)}{r_d s_d - r_q s_d \alpha}(c-1) + 1 \right) + h + 1 \right\rfloor$;
2 Poni neri tutti i nodi a profondità $\leq l'$;

In tempo $O(1)$ si può trovare la soluzione ottima, cioè sapere qual'è la profondità della frontiera. Il numero di nodi neri k utilizzati nella sozluione ottima si può trovare con l'equazione

$$k = \frac{c^{l'+1} - 1}{c-1}$$

Occorre poi tempo $O(k)$ per piazzare i nodi neri.
Si può trovare il costo energetico ottimo E in tempo $O(1)$ sommando i seguenti costi:

- $c^{l'} M_{l'} r_d s_d$: risalita fino al livello l' delle informazioni generate dai $c^{l'}$ sottoalberi radicati al livello l';

- $c^{l'}|T'_l|r_q s_d \alpha$: la risalita dal livello l' alla radice delle informazioni generate dagli alberi radicati al livello l';
- $R_{l'-1} r_q s_d \alpha$: risalita fino alla radice delle informazioni generate dai nodi con profondità al più $l'-1$, sottoforma di $R_{l'-1}$ messaggi di risposta;
- $\frac{c^{l'+1}-1}{c-1} r_q s_q$: ricezione (e invio) dell'interrogazione da parte dei $\frac{c^{l'+1}-1}{c-1}$ nodi con profondità al più l'.

Dove $M_{l'}$ è il numero di messaggi scambiati in un sottoalbero radicato al livello l'. Possiamo ricavare $M_{l'}$ dall'Equazione 2.1.3 a pagina 16 ponendo $i = l'$, ottenendo così:

$$M_{l'} = \sum_{j=1}^{h-l'} j c^j = \frac{c^{1-l'}\left(hc^{h+1} - l'c^{h+1} - c^{l'} - hc^h - (1-l')c^h\right)}{(c-1)^2}$$

Possiamo inoltre ricavare $R_{l'-1}$ grazie all'Equazione 2.1.4 a pagina 16:

$$R_{l'-1} = \sum_{j=1}^{l'-1} j c^j = \frac{ic^{1+l'} - (l')c^{l'} + c}{(c-1)^2}$$

Ricomponendo le somme possiamo trovare il costo ottimo E in tempo costante con l'equazione

$$E = c^{l'} M_{l'} r_d s_d + c^{l'}|T'_l|r_q s_d \alpha + R_{l'-1} r_q s_d \alpha + \frac{c^{l'+1}-1}{c-1} r_q s_q \tag{4.1.4}$$

Capitolo 5

Conclusioni

Nella tesi sono stati definiti gli algoritmi per trovare la disposizione ottima di un numero limitato e illimitato di nodi neri, in tre diversi modelli di comunicazione e per cinque diverse topologie di albero di comunicazione.

Riassumiamo ora la complessità di tutti gli algoritmi presentati. Nella Tabella 5.1 troviamo il costo per trovare la disposizione ottima dei nodi neri nei casi non limitati, mentre nella tabella 5.2 troviamo il medesimo costo per i casi limitati.

	Albero	Albero reg. comp.	Cammino	Caterpillar	Caterpillar reg.
Mod. 1	$O(n)$	$O(h)$	$O(1)$	$O(h)$	$O(1)$
Mod. 2	$O(n)$	$O(h)$	$O(1)$	$O(h)$	$O(h)$
Mod. 3	$O(n)$	$O(1)$	$O(1)$	$O(h)$	$O(h)$

Tabella 5.1: *Costo per individuare la soluzione ottima nel caso **non limitato**. Ogni riga esamina un diverso modello di comunicazione. In ogni colonna troviamo una topologia diversa, nell'ordine: albero, albero regolare completo, cammino, caterpillar, caterpillar regolare.*

	Albero	Albero reg. comp.	Cammino	Cat.	Cat. reg.
Mod. 1	$O(kn^2(\max\{k,c\})^{c-1})$	$O\left(k(\log^2 n)(\max\{k,c\})^{c-1}\right)$	$O(\log k)$	$O(h^2k)$	$O(\min\{h,k\})$
Mod. 2	$O(kn^2(\max\{k,c\})^{c-1})$	$O\left(k(\log^2 n)(\max\{k,c\})^{c-1}\right)$	$O(\log k)$	$O(h^2k)$	$O(\min\{h,k\})$
Mod. 3	$O(kn^2(\max\{k,c\})^{c-1})$	$O\left(k(\log^2 n)(\max\{k,c\})^{c-1}\right)$	$O(\log k)$	$O(h^2k)$	$O(\min\{h,k\})$

Tabella 5.2: *Costo per individuare la soluzione ottima nel caso **limitato**. Ogni riga esamina un diverso modello di comunicazione. In ogni colonna troviamo una topologia diversa, nell'ordine: albero, albero regolare completo, cammino, caterpillar, caterpillar regolare.*

Osserviamo anzitutto che per tutte le topologie considerate, e per tutti i modelli di comunicazione, il costo per risolvere il caso non limitato è sempre inferiore al caso limitato. In tutte queste situazioni era dunque valida l'assunzione che l'algoritmo per il caso non limitato avesse costo inferiore al caso limitato, e quindi assumere $k < k_u$ non aumentava la complessità degli algoritmi nel caso limitato.

Per quanto riguarda il solo caso limitato i costi degli algoritmi sul primo modello sono identici a quelli per il secondo modello. Ciò fa presumere che la caratteristica del primo mo-

dello, cioè inviare sempre l'interrogazione a tutti i figli, non sia una semplificazione efficace per abbassare il costo degli algoritmi. Si noti inoltre che nel caso illimitato tale semplificazione abbassa il costo di un solo algoritmo, su grafo caterpillar regolare.

Come visto nella sezione 4 gli algoritmi per il Modello 2 possono essere usati per risolvere il problema sul Modello 3: questo implica che risolvere il problema del piazzamento delle memorie nel Modello 2 è intrisicamente complicato almeno quanto risolverlo nel Modello 3. Il costo degli algoritmi specifici per il Modello 3 inoltre dà un limite inferiore per il costo per gli algoritmi sul Modello 2.

Sviluppi futuri

La tesi fornisce vari spunti su come proseguire gli studi in merito al piazzamento delle memorie in un modello ad albero fisso. Cercare nuovi algoritmi con costo inferiore per i casi già trattati è solo la prima possibilità; qui di seguito presentiamo una lista di possibili approfondimenti.

- Sarebbe interessante studiare l'andamento del risparmio energetico all'aumentare del numero di memorie disponibili, per le diverse topologie, cercarndo di scoprire da cosa esso dipenda.
- Nel caso limitato su albero generico, la maggior parte del costo computazionale è dovuta al determinare come spartire i nodi neri disponibili tra i sottoalberi: sarebbe dunque utile investigare se questa suddivisione dipende da qualche caratteristica dell'albero stesso.
- È possibile utilizzare il piazzamento delle memorie per il risparmio energetico anche con metriche diverse. Per esempio considerare non il costo enegetico totale ma il costo che incorre in ogni singolo nodo, in modo da aumentare il tempo che trascorre prima della disconnessione di un nodo. Proseguendo in questa direzione, si potrebbe prendere come metrica da minimizzare il massimo tra i costi che incorrono in ogni nodo

$$\max_{\forall i \in T} \{e(i)\}$$

- In molte reti sensoriali non cablate, la topologia iniziale dell'albero di comunicazione può variare anche frequentemente, sarebbe quindi opportuno prevedere tale circostanza e studiare una disposizione delle memorie che ne tenga conto.
- Si può studiare il piazzamento di memorie in un modello di rete di sensori cablata adattando gli algoritmi sviluppati per il terzo ed il secondo modello.

Bibliografia

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] M. Matin and M. Islam, "Overview of wireless sensor network," *Wireless Sensor Networks - Technology and Protocols*, 2012. Disponibile presso `http://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/overview-of-wireless-sensor-network`.

[3] "Wireless sensor network," 2012. `http://en.wikipedia.org/wiki/Wireless_sensor_network`.

[4] M. Ilyas and I. Mahgoub, *Handbook of sensor networks: compact wireless and wired sensing systems*. CRC, 2004.

[5] Y. Wang, X. Fu, and L. Huang, "A storage node placement algorithm in wireless sensor networks," in *Frontier of Computer Science and Technology, 2009. FCST'09. Fourth International Conference on*, pp. 267–271, IEEE, 2009.

[6] C. S. Raghavendra, K. M. Sivalingam, and T. Znati, *Wireless sensor networks*. Springer, 2006.

[7] B. Sheng, Q. Li, and W. Mao, "Optimize storage placement in sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 10, pp. 1437–1450, 2010. Disponibile presso `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.172.7298&rep=rep1&type=pdf`.

[8] Z. Eskandari and F. Ayughi, "Data aggregation tree construction: Algorithms and challenges,"

[9] B. Sheng, C. Tan, Q. Li, and W. Mao, "An approximation algorithm for data storage placement in sensor networks," in *Wireless Algorithms, Systems and Applications, 2007. WASA 2007. International Conference on*, pp. 71–78, IEEE, 2007. Disponibile presso `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.122.9352&rep=rep1&type=pdf`.

[10] S. A. Aly, A. Ali-Eldin, and H. V. Poor, "A distributed data collection algorithm for wireless sensor networks with persistent storage nodes," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pp. 1–5, IEEE, 2011.

[11] "Power sum," 2012. `http://mathworld.wolfram.com/PowerSum.html`.

[12] F. Harary and A. Schwenk, "The number of caterpillars," *Discrete Mathematics*, vol. 6, no. 4, pp. 359–365, 1973. `http://hdl.handle.net/2027.42/33977`.

[13] D. Knuth, "The art of computer programming, fascicle 3: Generating all combinations and partitions, vol. 4," 2005.