Difficoltà e Target

Mining di un blocco in Bitcoin

Indice generale

Premessa

Questi appunti contengono alcuni appunti sui concetti di difficoltà e target nel contesto dell'aggiunta di un nuovo blocco alla blockchain di Bitcoin. Il documento assume per acquisiti i concetti elementari della tecnologia Bitcoin (SHA256, notazione esadecimale ecc.).

Definizione di target

Affinché un nuovo blocco venga "minato, chi lo produce deve trovare un *nounce* tale che, una volta aggiunto al blocco stesso, faccia sì che l'hash dell'intero blocco (ovvero: hash del blocco precedente, insieme di tutte le transazioni, *nouce* prodotto ad hoc) sia **minore** di un certo numero: tale numero è detto **target** del processo di mining.

Esempio: la funzione di hash (SHA256) produce un hash a 32 byte, ovvero un numero di 256 bit, compreso tra 0 e $1{,}16 \cdot 10^{77}$ (circa). Perciò, se il target attuale fosse pari a $5 \cdot 10^{42}$, allora un hash di $6 \cdot 10^{42}$ sarebbe invalido (perché maggiore del target) mentre un hash di $4 \cdot 10^{42}$ sarebbe valido (perché minore del target).

Il target viene spesso espresso in forma esadecimale, ad esempio:

00 00 00 00 00 00 00 FA 56 7D 8B 84 … … … C6 70 E4 72 (32 byte)

Per cui, minore è il target, più "zeri" ci saranno all'inizio della sua rappresentazione esadecimale. Ecco perché, per spiegare il concetto in maniera intuitiva, spesso si dice che l'hash di un blocco è valido se contiene al massimo "un certo numero di zeri iniziali".

A rigore ciò non è esatto, ma dal punto di vista divulgativo tale affermazione è accettabile, perché illustra in modo semplice il concetto, al prezzo di una piccola imprecisione. Approfondiremo quest'aspetto nei prossimi capitoli.

Target espresso in Bits

Il valore del parametro **Bits** è un modo alternativo di indicare il valore del target. Come nel caso della temperatura, che può essere espressa in gradi Celsius, Kelvin o Fahrenheit, anche il target può essere espresso in diverse "unità di misura". Per capire il significato del parametro Bits è utile analizzare il primo blocco della blockchain (il cosiddetto *genesis block*).

Il target e l'hash del *genesis block* sono:

- Bits: 486,604,799
- Hash:000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

Vediamo come convertire la misura del target espresso in Bits nel corrispondente valore esadecimale. Come prima cosa dobbiamo precisare che Bits è un numero di 4 byte, la cui rappresentazione decimale (in questo caso 486,604,799) è poco utile. Se invece convertiamo il numero 486,604,799 in esadecimale, allora troviamo i 4 byte che ci servono:

Bits = 1D 00 FF FF

Che vanno letti usando il seguente **formato**:

- Primo byte: indica il numero di byte totali del target
- Altre tre byte: coincidono con le prime tre cifre significative del target

Nell'esempio qui sopra il target è composto da un numero di 29 byte (1D in esadecimale) le cui cifre più significative sono proprio "00 FF FF". Se esprimiamo tale numero utilizzando tutti e 32 i byte previsti dal formato, il target in Bits si traduce nel seguente target esadecimale:

00 00 00 00 FF FF 00 00 00 … 00 00 00 00 (in verde i 29 byte del target)

che convertito in decimale corrisponde a:

$$\textbf{Target}|_{\textbf{GENESIS}} = 255 \cdot 256^{27} + 255 \cdot 256^{26} + 0 \cdot 256^{25} + \ldots + 0 \cdot 256^{0} \approx \mathbf{2{,}7 \cdot 10^{67}}$$

Se convertiamo i parametri del *genesis block* in decimale abbiamo quindi:

- Target: $2{,}7 \cdot 10^{67}$ (rappresentazione decimale del target espresso come Bits = 486,604,799)
- Hash: $1{,}6 \cdot 10^{64}$ (corrispondente all'hash 000000000019d6689 … ce26f del primo blocco)

Si vede che l'hash è molto minore del target, il che conferma che il blocco è costruito in modo corretto. E' curioso notare che l'hash sarebbe stato valido anche se avesse avuto un paio di "zeri iniziali" in meno (ad esempio: 000000FF19d6689...). Infatti in tal caso si avrebbe avuto un hash pari a $2{,}68 \cdot 10^{67}$, comunque minore del target (pari a $2{,}7 \cdot 10^{67}$).

Numero di zeri iniziali

Spieghiamo adesso perché viene detto che "minare un blocco" significa trovare un hash "con almeno un certo numero di zeri". Partiamo dalla definizione corretta, ovvero:

l'hash del blocco è valido <=> hash ≤ target

dove, per il **genesis block**, l'ultima disuguaglianza è:

$1{,}6 \cdot 10^{64} \leq 2{,}7 \cdot 10^{67}$

la stessa disuguaglianza, in notazione esadecimale, diventa:

Target :	00 00 00 00 FF FF 00 00 00 00 00 … 00 00 00 00 00	(32 byte)
Hash:	00 00 00 00 00 19 d6 68 9c 08 5a ... 8c e2 6f	(32 byte)

In questo caso "si vede ad occhio" che il numero "00 19 d6 68 ..." è **minore** di "00 FF FF 00 ...". Infatti basta osservare che il primo byte significativo dell'hash è il 6° da sinistra, mentre il primo byte significativo del target è in 5° posizione. Questo è del tutto analogo a come si ragiona in notazione decimale, dove il numero *007* è minore del numero *042* semplicemente perché il primo è un numero con una sola cifra significative ($\sim 10^1$), mentre il secondo ha due cifre significative ($\sim 10^2$).

Nota: quanto appena detto è del tutto equivalente ad affermare che *7* è minore di *42*. Abbiamo anteposto uno zero iniziale, scrivendo *007* e *042*, solo per sottolineare l'analogia tra notazione decimale e notazione esadecimale. In ambo i casi gli zeri iniziali possono essere ignorati, perché sono cifre *non* significative. La scelta di specificare un certo numero di zeri iniziali risulta comodo nell'ambito informatico, dove a volte è necessario visualizzare senza ambiguità la posizione delle cifre nella "cornice" dei 32 byte possibili.

Il confronto visivo tra target ed hash spiega perché, allo scopo di semplificare la divulgazione dell'argomento in oggetto, è comodo dire che un hash è valido se "inizia con un numero di zeri minore del target". Si tratta di una piccola imprecisione, che nella maggior parte dei casi è comunque verificata. Infatti, se un hash inizia con meno zeri del target, allora è evidente che l'hash sia minore del target. In altre parole dire che "un hash è valido se inizia con un numero di zeri minore del target" è una condizione sufficiente ma non necessaria. Infatti è possibile avere *hash* ≤ *target* anche se i due numeri hanno lo stesso numero di zeri.

Difficoltà

Abbiamo visto come il target del mining possa essere definito in diverse "unità di misura": come numero esadecimale (32 byte), decimale (fino a $1{,}16 \cdot 10^{77}$), oppure in rappresentazione **compatta**, ovvero in **Bits**. In ogni caso, qualsiasi sia il formato scelto, il target rappresenta sempre il limite superiore ammissibile per l'hash del prossimo blocco.

Ad esempio, il target del *genesis block* può essere espresso come:

- Decimale: $2{,}7 \cdot 10^{67}$ (circa)
- Esadecimale: 00 00 00 00 FF FF 00 00 … 00 00 00 00 (32 byte)
- Bits: 486,604,799 (ovvero 1D 00 FF FF)

Un altro modo di esprimere il target è per mezzo della **difficoltà**, così definita:

Difficoltà = Target(1° blocco) / Target(corrente)

Bitcoin è progettato in modo che il target massimo sia quello del primo blocco, per cui l'algoritmo di *re-targeting* (spiegato più sotto) è in grado di aumentare o diminuire il target, ma esso sarà sempre e comunque maggiore di quello del primo blocco. Ciò suggerisce di introdurre una nuova unità di misura, ossia di definire la difficoltà del primo blocco pari ad **uno** (imponendo che la difficoltà degli altri blocchi non sia mai inferiore ad uno):

Target(massimo) = Target del primo blocco (per definizione)

per cui possiamo scrivere:

Target(corrente) = Target(1° blocco) / Difficoltà $\approx 2{,}7 \cdot 10^{67}$ / Difficoltà

Esempio 1

Sapendo che al 1° marzo 2022 la difficoltà era pari a circa $28 \cdot 10^{12}$ (per la precisione: 27,967t, dove *t* = *Tera* = 10^{12}) si ottiene [4]:

$$\text{Target}|_{\text{marzo 2022}} \approx 2{,}7 \cdot 10^{67} / 28 \cdot 10^{12} \approx 9{,}6 \cdot 10^{53}$$

che in notazione esadecimale diventa (23 byte):

0A 05 DA A7 B8 0E F2 04 E5 5F B6 C3 43 2A 12 26 00 00 00 00 00 00 00

per esprimere questo numero nella notazione a 32 byte, aggiungiamo i 9 byte mancanti:

00 00 00 00 00 00 00 00 00 0A 05 DA A7 B8 0E F2 04 E5 5F B6 C3 43 2A 12 26 00 00 00 00 00 00 00

Si potrebbe quindi dire che, in prima approssimazione, al 1° marzo 2022 gli hash validi erano quelli che iniziavano con 9 byte pari a 00, cioè con 18 zeri. Verifichiamo questa affermazione, confrontando il target scritto qui sopra con il valore del parametro *Bits* per il blocco 725'355 (del 1° marzo 2022) [5]:

Hash: 00000000000000000000481ef0e1fad756239f3bae753c129bbb91d098f1ebfba
Difficoltà: 27'967'152'532'434,23 (ovvero quasi $28 \cdot 10^{12}$)
Bits: 386,535,544 (in esadecimale: 17 0A 10 78)

Come visto sopra, il valore espresso in Bits ci dice che il target inizia con "0A 10 78" e contiene 23 byte significativi (la prima cifra di Bits, cioè 17 in esadecimale) ovvero:

00 00 00 00 00 00 00 00 0A 10 78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

che in decimale corrisponde ad un target pari a circa $9{,}6 \cdot 10^{53}$, da cui si ottiene:

$$\text{Difficoltà} \approx 2{,}7 \cdot 10^{67} / 9{,}6 \cdot 10^{53} \approx 28 \cdot 10^{12}$$

che è proprio la difficoltà associata al blocco. Questo conferma quanto detto relativamente ai criteri di conversione tra target espresso nelle quattro notazioni discusse: numero decimale, numero esadecimale, Bits (4 byte) o difficoltà.

Esempio 2

Stimiamo adesso la probabilità di minare un blocco valido, conoscendo la difficoltà dell'operazione. Ricordiamo che per definizione vale:

Target $\approx 2{,}7 \cdot 10^{67}$ / Difficoltà

poiché al 1° Marzo 2022 la difficoltà era pari a $28 \cdot 10^{12}$, si ha (come già visto):

$\text{Target}|_{\text{marzo 2022}} \approx 2{,}7 \cdot 10^{67} / 28 \cdot 10^{12} \approx 9{,}6 \cdot 10^{53}$

Sapendo che l'hash prodotto dall'algoritmo SHA256 è un numero a 32 byte, il processo di mining può essere visto come l'estrazione di un numero a caso compreso da 0 e $1{,}16 \cdot 10^{77}$. Affinché un blocco sia valido il numero estratto deve essere *minore* del target, per cui la **probabilità** *P* di ottenere un hash valido è:

$P(\text{hash valido}) = P(\text{hash} < \text{target}) = 9{,}6 \cdot 10^{53} / 1{,}16 \cdot 10^{77} \approx 8{,}3 \cdot 10^{-24}$

Assumendo di minare con uno dei migliori prodotti al tempo di questo scritto (Giugno 2022), si avrebbero circa 18 milioni di MHash/s, pari a $1{,}8 \cdot 10^{13}$ hash al secondo, per cui il *valore atteso* del tempo per minare un blocco valido sarebbe [6]:

$\Delta t = 1 / (\text{Hashing rate} \cdot \text{probabilità}) = 1 / (1{,}8 \cdot 10^{13} \cdot 8{,}3 \cdot 10^{-24}) \approx 6{,}7 \cdot 10^{9} \approx 212$ anni

Vale a dire: anche possedendo un centinaio dei miner più efficaci, il tempo medio di produzione di un singolo blocco sarebbe dell'ordine degli anni. Chiaramente ciò non è un problema, perché l'hash rate totale dell'intera rete Bitcoin è molto più elevato. Ad esempio, nel marzo 2022 l'hash rate della rete Bitcoin era pari a circa 195m TH/s [7], da cui:

$\text{Network hashrate} = 195 \cdot 10^{6} \cdot 10^{12} \approx 2 \cdot 10^{20} \approx 200 \cdot 10^{18} \approx 200$ EH/s

per cui il tempo medio di attesa di produzione di un blocco era:

$\Delta t = 1 / (\text{Hashing rate} \cdot \text{probabilità}) = 1 / (2 \cdot 10^{20} \cdot 8{,}3 \cdot 10^{-24}) \approx 602 \approx 10$ minuti

esattamente in linea con il tempo di attesa voluto da Satoshi Nakamoto nella primissima versione del protocollo Bitcoin.

Re-targeting

Come è noto, circa ogni due settimane circa la rete Bitcoin aggiorna la difficoltà. In realtà ciò avviene esattamente ogni **2016** blocchi. Infatti, nello scenario ideale in cui un blocco venisse generato ogni 10 minuti, 2016 blocchi corrisponderebbero a :

$$T = 2016 \cdot 10 \text{ min} = 20160 \text{ min} = 14 \text{ giorni}$$

Perciò, quando si dice che "Bitcoin è progettato per aggiornare la difficoltà ogni 2 settimane", in realtà si intende che ogni 2016 blocchi si calcola il parametro K:

$$K = 20160 / (\text{minuti impiegati per generare gli ultimi 2016 blocchi})$$

quindi K < 1 se la blockchain sta crescendo troppo lentamente, K > 1 se è troppo rapida.
Di conseguenza la difficoltà per il prossimo blocco viene reimpostata come segue:

$$\textbf{Difficoltà}|_{\textbf{nuova}} = K \cdot \text{Difficoltà}|_{\text{vecchia}} \qquad \text{(proporzione lineare tra K e difficoltà)}$$

in questo modo, se la generazione dei nuovi blocchi rallenta (K < 1), la nuova difficoltà sarà *minore* della precedente, realizzando così il processo di controllo automatico (retroazione negativa) che stabilizza il sistema. Questo è il meccanismo che "garantisce" che venga minato un blocco valido in media ogni 10 minuti, dove le virgolette servono ad enfatizzare che tale garanzia non è realizzata tramite un feedback in tempo reale, ma solo ogni 2 settimane circa.
E' quindi normale che ci sia un lieve ritardo tra la cambiamento di velocità nella generazione dei blocchi, e il relativo aggiustamento della difficoltà.

Aspetti termodinamici

Dal punto di vista termodinamico si può pensare che un dispositivo di mining sia un sistema di cui sia noto solo il macrostato. Poiché un miner può generare un numero compreso tra 1 e $1{,}16 \cdot 10^{77}$, è lecito pensare che a tale macrostato fisico (equilibrio termico) corrispondano circa 10^{77} microstati diversi, equivalenti tra loro. Una piccola parte di questi microstati appartiene però anche ad una altro macrostato, vale a dire al macrostato in cui l'hash prodotto è minore del target (e quindi il blocco è valido). Possiamo perciò scrivere:

- Macrostato "neutro": un qualsiasi microstato tra i 10^{77} possibili
- Macrostato "blocco valido": un qualsiasi microstato tale da avere hash ≤ target

L'informazione media associata al macrostato "neutro" vale [8]:

$$H_0 = \langle I(x) \rangle = - \sum P_x \cdot \log_b(P_x) = - \log_b(10^{-77}) \cdot \sum P_x = - \log_b(10^{-77}) = \log_b(10^{77})$$

scegliendo come base *b = 2* (informazione misurata in *bits*) e ricordando (proprietà dei logaritmi) che vale $\log_2(x) \approx \log_{10}(x) \cdot 3{,}32$ si ottiene:

$$H_0 = \langle I(x) \rangle = \log_2(10^{77}) \approx \log_{10}(10^{77}) \cdot 3{,}32 \approx 256 \text{ bit}$$

com'era logico aspettarsi, visto che lo SHA256 produce segnali proprio a 256 bit. L'informazione media associata al macrostato "blocco valido" (al 1° marzo 2022) è invece:

$$H_1 = \langle I(x) \rangle \approx \log_{10}(9{,}6 \cdot 10^{53}) \cdot 3{,}32 \approx 179 \text{ bit}$$

Per cui l'estrazione di un hash valido può essere pensata come una riduzione dell'entropia del sistema, come se il sistema passasse dal macrostato "neutro" a quello "valido".

Ricordando che si ha [9]:

$$\mathbf{S} = k_B \cdot \ln(2) \cdot H$$ ($K_B = 1{,}38 \cdot 10^{-23}$ è la costante di Boltzmann)

la variazione di entropia associato alla trasformazione *neutro → valido* vale:

$$\Delta S = S_1 - S_0 = k_B \cdot \ln(2) \cdot (H_1 - H_0) \approx 10^{-23} \cdot (179 - 256) \approx - 10^{-21} \text{ J/K}$$

Il risultato ci dice che, almeno in linea teorica, ovvero assumendo di lavorare con dispositivi reversibili (computazione reversibile) e di consumare energia solo in fase di cancellazione della memoria (principio di Landauer), l'energia **minima** necessaria per minare un blocco valido a temperatura ambiente è (al 1° Marzo 2022) [10]:

$$\mathbf{E_{min}} = T \cdot |\Delta S| \approx 300 \cdot |10^{-21}| \approx \mathbf{10^{-18}\ Joule}$$

Siccome l'energia consumata da un dispositivo di mining è di svariati ordini di grandezza maggiore di tale limite teorico, il risultato ci dice solo che a livello tecnologico siamo ancora lontanissimi dai consumi teoricamente ottenibili nel pieno rispetto delle leggi fisiche. Un margine così elevato lascia però sperare che in futuro sia possibile minare in modo più efficiente, con gli ovvi benefici economici ed ambientali.

Bibliografia

[1] Blockchain.com, *Genesis Block*, 2022
https://www.blockchain.com/btc/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

[2] Rapid Tables, *Hexadecimal to Decimal converter*, 2022
https://www.rapidtables.com/convert/number/hex-to-decimal.html

[3] Inch Calculator, *Scientific Notation Calculator and Converter*, 2022
https://www.inchcalculator.com/scientific-notation-calculator/

[4] Blockchain.com, *Network Difficulty*, 2022
https://www.blockchain.com/charts/difficulty

[5] Blockchain.com, *Blocco* 725355, 2022
https://www.blockchain.com/btc/block/725355

[6] Bitcoin Wiki, *Mining hardware comparison*, 2022
https://en.bitcoin.it/wiki/Mining_hardware_comparison

[7] Blockchain.com, *Total Hash Rate (TH/s)*, 2022
https://www.blockchain.com/charts/hash-rate

[8] Stefano Adriani, *Teoria dell'Informazione*, 2022
http://adriani.altervista.org/author/notes.php

[9] Stefano Adriani, *Informazione e Termodinamica*, 2022
http://adriani.altervista.org/author/notes.php

[10] Stefano Adriani, *Il Principio di Landauer*, 2022
http://adriani.altervista.org/author/landauer_01.php