Rendiconti
Accademia Nazionale delle Scienze detta dei XL
Memorie e Rendiconti di Chimica, Fisica, Matematica e Scienze Naturali
139° (2021), Vol. II, fasc. 3, pp. 179-183
ISSN 0392-4130 • ISBN 978-88-98075-48-5

Randomness e Pseudorandomness nella Computazione*

LUCA TREVISAN

Università Bocconi, Milano
E.mail: l.trevisan@unibocconi.it

Abstract – Theoretical computer science concerns itself with the rigorous analysis of algorithms (from the points of view of correctness and efficiency) and with the study of the limits of computation, with applications to cryptography and computer security. Furthermore, mathematical techniques developed for the analysis of algorithmic processes have found applications in the sciences and in pure mathematics.

In these notes we will outline the contributions of theoretical computer science to the understanding of the role of randomness in computation. We will trace the development of several ideas, from the 1950s to the 1980s, on the use of probabilistic methods to design algorithms for purely deterministic problems, and the recognition, also in the 1980s, of the fundamental role played by probabilistic methods in the development of secure cryptosystems. We will then see how another thread of research developments has provided increased evidence that all probabilistic algorithms admit efficient deterministic simulations, and see how a recent outcome of these developments has been the resolution of a long-standing open problem in combinatorics.

This review will show how the key goals of theoretical computer science (of designing provably efficient and correct algorithms; of understanding the limitations of efficient algorithms; and of developing mathematical techniques of broader applicability) are intertwined.

Keywords: Theoretical Computer Science, Computational Complexity, Derandomization, Pseudorandom Generators, Randomness Extraction

1. Introduzione

L'informatica teorica studia i fondamenti matematici della computazione, e tra i suoi obiettivi principali, c'è lo sviluppo di tecniche per progettare algoritmi efficienti, per dimostrarne la correttezza, e per studiare il modo in cui il tempo di calcolo e altre risorse scalano al crescere dei dati da elaborare.

Un obiettivo complementare è quello di studiare i *limiti* della risolubilità algoritmica di problemi, e di identificare problemi che non ammettono soluzioni

* Prolusione tenuta in occasione dell'Inaugurazione del 240° Anno Accademico dell'Accademia Nazionale delle Scienze detta dei XL, tenutasi il 19 maggio 2022 presso la biblioteca dell'Accademia.

algoritmiche efficienti. Questo secondo obiettivo, che definisce lo studio della *complessità computazionale*, ha importanti ramificazioni applicative, in particolare allo sviluppo di tecniche crittografiche e alla sicurezza informatica. Quasi sempre, infatti, la sicurezza di sistemi crittografici può essere violata da un avversario che abbia tempo di calcolo illimitato e che possa, per esempio, enumerare tutte le possibili chiavi segrete di cifratura fino a trovare quella giusta. Un tale attacco di «forza bruta», richiederebbe però miliardi di anni anche per chiavi di cifratura di lunghezza modesta, e la sicurezza di tali sistemi si basa sull'evidenza che non esistano attacchi più efficienti. Lo studio di problemi computazionali che siano risolubili in tempo finito, ma necessariamente in tempo che cresca esponenzialmente con i dati da elaborare, è quindi centrale all'analisi di sicurezza dei sistemi crittografici.

Le tecniche matematiche sviluppate per l'analisi di algoritmi e lo studio della complessità computazionale hanno trovato applicazione sia nella modellazione di problemi nelle science fisiche e naturali sia nella risoluzione di problemi di matematica pura, per esempio la recente dimostrazione della congettura di Kadison e Singer [9] usando tecniche sviluppate nell'ambito degli algoritmi su reti o la refutazione della «embedding conjecture» di Connes [8] usando tecniche sviluppate nell'ambito degli algoritmi quantistici.

Le summenzionate tre direttrici di ricerca (lo sviluppo di algoritmi efficienti, lo studio della complessità computazionale, e le applicazioni ad altre scienze) sono spesso intimamente legate. In queste note tracceremo alcuni sviluppi dello studio della randomness nella computazione dagli anni '50 ad oggi, mostrando come l'evoluzione di idee relative allo sviluppo di algoritmi, alla sicurezza crittografica e alla complessità computazionale si siano intrecciate, portando una nuova prospettiva sulle computazioni probabilistiche e ad innovative ricadute in matematica pura.

2. Lo sviluppo di algoritmi probabilistici

Tecniche probabilistiche possono essere usate con successo per risolvere problemi di natura puramente deterministica. Un esempio classico è quello di stimare il valore di π nel modo seguente: se inscriviamo un cerchio in un quadrato, allora $\frac{\pi}{4}$ è uguale alla probabilità che, scegliendo un punto a caso nel quadrato, tale punto appartenga anche al cerchio. Quest'ultima probabilità può essere stimata empiricamente scegliendo una sequenza di punti a caso nel quadrato e calcolando la frazione di tali punti che cadono nel cerchio. Questa procedura si può tradurre in un algoritmo: scegliamo una sequenza casuale $(x_1, y_1), \ldots, (x_k, y_k)$ tale che ogni x_i ed ogni y_i siano uniformemente distribuiti tra -1 e $+1$, sia t il numero di coppie (x_i, y_i) tale che $x_i^2 + y_i^2 \leq 1$, e diamo in output il valore $4t/k$. Al crescere di k, con alta probabilità tale algoritmo calcola una stima accurata di π. Ci sono metodi deterministici molto più efficienti per stimare π numericamente, ma questo esempio mostra come un problema di calcolo puramente deterministico possa essere affrontato con metodi probabilistici, creando un processo stocastico tale che una quantità facile da stimare del processo stocastico (nel caso di cui sopra, il numero medio di punti scelti nel quadrato che cadono nel cerchio inscritto) sia la risposta al problema di interesse (stimare π).

Una delle prime applicazioni dei metodi probabilistici allo sviluppo di algoritmi è stata la tecnica *Markov Chain Monte Carlo* (abbreviata MCMC) di Metropolis e Hastings [10, 6]. Tale algoritmo consente di campionare da certe distribuzioni di probabilità che si incontrano nella fisica statistica, e trova molte applicazioni. In una delle tipiche applicazioni di questa tecnica, abbiamo un problema di ottimizzazione discreto, per esempio un problema di logistica, di ricerca operativa, o di gestione aziendale, in cui c'è un insieme X noto ma molto grande (esponenzialmente grande nella quantità di dati nota) di possibili soluzioni al problema, tale che ogni soluzione $x \in X$ ha un costo $f(x)$, e vogliamo trovare un $x^* \in X$ che minimizzi il costo $f(x^*)$.

Un approccio MCMC a un problema di ottimizzazione discreta è di associargli (come esperimento mentale) un sistema fisico in cui ogni soluzione x è un possibile stato del sistema, e $f(x)$ è la sua energia. Per tale sistema, la distribuzione di Gibbs a temperatura T è una distribuzione di probabilità su X tale che la probabilità di ogni possibile stato x è proporzionale a $e^{-\frac{1}{T}f(x)}$. A basse temperature, le soluzioni $x \in X$ del problema di ottimizzazione che minimizzano $f(x)$ dominano la distribuzione, perciò la capacità di campionare dalla distribuzione di Gibbs a basse temperature è sostanzialmente equivalente alla risoluzione del problema di ottimizzazione. L'algoritmo di Metropolis-Hastings dà un metodo per campionare da distribuzioni di Gibbs in modo approssimato, e quindi di trovare soluzioni ottimali o approssimate per il problema di ottimizzazione di interesse.

Anche in questo caso, il passaggio da un problema deterministico ad un algoritmo probabilistico avviene associando al problema un processo stocastico tale che una proprietà del processo stocastico (in questo caso, gli

stati che hanno probabilità più alta nella distribuzione di Gibbs) è associata alla soluzione corretta del problema deterministico.

Un'applicazione ancora più soprendente di tecniche probabilistiche negli algoritmi fu trovata per il problema di determinare se un dato numero intero è primo. Per esempio, supponiamo che ci sia dato il numero

$$2355509180448628135 7$$

e che si voglia determinare se è un numero composto, cioè un prodotto di due interi più piccoli diversi da uno, oppure se è primo, cioè privo di divisori diversi da uno e da se stesso. Un approccio potrebbe essere di provare tutti i divisori possibili, e verificare se la divisione possa essere eseguita senza resto. Un tale approccio, però, richiede di enumerare tutti gli interi minori del numero dato (o, ottimizzando leggermente, minori della radice quadrata del numero dato), e il numero di casi da considerare cresce esponenzialmente con il numero di cifre del numero dato, e già con l'esempio di cui sopra è un numero di casi molto grande.

Possiamo però usare il cosiddetto «piccolo teorema di Fermat» che afferma che se un numero n è primo, allora per ogni intero x abbiamo l'equazione

$$x^{n-1} \equiv 1 \pmod{n}$$

Prendendo n come il numero di cui sopra, e, per esempio $x = 5$, possiamo calcolare che

$$5^{2355509180448628135 6} \equiv 4974600172144501056 \pmod{2355509180448628135 7}$$

che dimostra che $n = 2355509180448628135 7$ non può essere primo. Questo suggerisce la seguente euristica per verificare se un dato numero n è composto: scegliere una base x a caso, e verificare se l'equazione $x^{n-1} \equiv 1 \pmod{n}$. Se si trova una violazione del piccolo teorema di Fermat, allora n è sicuramente composto; se non si trova alcuna violazione dopo più tentativi, si può ipotizzare con una certa confidenza che n sia primo? Questo approccio non funzionerebbe, ma alla fine degli anni '70, Miller e Rabin [11, 14] e Solovay e Strassen [15] dimostrarono che esistono delle altre equazioni simili a quella che definisce il piccolo teorema di Fermat e tali che, per ogni numero composto, c'è un'alta probabilità di trovare un'equazione violata semplicemente scegliendola a caso. Gli algoritmi di Miller-Rabin e Solovay-Strassen danno quindi metodi probabilistici dimostrabilmente validi per verificare, con alta probabilità, se un dato numero è primo o composto. Il tempo di calcolo di tali algoritmi cresce in modo polinomiale rispetto al numero di cifre dell'intero dato, e quindi è applicabile efficientemente anche ad interi con migliaia o decine di migliaia di cifre, cioè numeri interi così grandi che se provassimo ad enumerarne i possibili divisori avremmo molti più casi da considerare che il numero di atomi nell'universo.

Gli algoritmi di Miller, Rabin, Solovay e Strassen provocarono un'esplosione di interesse nello sviluppo di algoritmi probabilistici con garanzie dimostrabili di correttezza ed efficienza, e tali algoritmi vennero sviluppati negli ambiti più diversi, compresi gli algoritmi su reti, l'ottimizzazione continua e discrete, e lo sviluppo di strutture dati. Una rassegna dei risultati ottenuti in questo campo alla metà degli anni '90 è presente nella monografia di Motwani e Raghavan [12], e molte altre applicazioni sono state sviluppate nei decenni seguenti.

I successi nello sviluppo di algoritmi probabilistici pongono la seguente questione aperta:

Se un problema computazionale è risolubile con un algoritmo probabilistico che ha dimostrabilmente alta probabilità di trovare una soluzione corretta ed un tempo di calcolo efficiente, allora tale problema computazionale è anche sempre risolubile con un algoritmo deterministico di simile efficienza e che trovi sempre una soluzione corretta?

In altre parole, è sempre possibile evitare l'uso di tecniche probabilistiche, e quindi avere zero probabilità di errore, senza che ciò porti a tempi di calcolo significativamente maggiori? Questa è una delle questioni aperte fondamentali della teoria della complessità computazionale. Alla fine degli anni '80, l'evidenza portava ad ipotizzare una risposta negativa, ma ricerche condotte negli anni '90 hanno dato evidenza molto forte che, almeno per una definizione appropriata di «simile efficienza», la risposta sia positiva. A conferma di questa predizione, un algoritmo deterministico efficiente per determinare se un dato intero è primo è stato sviluppato nel 2004 [1] (in precedenza, il problema della primalità era un esempio di problema computazionale per il quale esisteva un gap esponenziale tra l'efficienza del miglior algoritmo probabilistico noto rispetto al miglior algoritmo deterministico noto).

3. Lo sviluppo della crittografia moderna

Gli studiosi di complessità computazionale ritengono quindi che sia sempre possibile eliminare le tecniche probabilistiche dal progetto di algoritmi, ad un costo relativamente contenuto dal punto di vista delle prestazioni. Le tecniche che hanno portato a queste conclusioni

provengono dallo sviluppo della crittografia moderna che, ironicamente, è un'area dell'informatica nella quale le tecniche probabilistiche sono invece assolutamente *necessarie*.

La crittografia nasce, in tempi antichi, come studio delle tecniche per cifrare messaggi in modo che non possano essere decodificati se intercettati, in assenza di chiavi di decodifica segreta. La crittografia moderna è alla base del funzionamento dell'economia digitale, e oltre a fornire soluzioni per la trasmissione e la conservazione di dati sensibili, e per la firma digitale, consente la creazione di sistemi distribuiti sicuri, sia nel caso di sistemi centralizzati sia nel caso di sistemi decentralizzati come le blockchain.

In tutte le applicazioni della crittografia, compresa quella più tradizionale della cifratura di messaggi sensibili, l'uso di tecniche probabilistiche è necessario perché, se le chiavi di cifratura non sono scelte in modo casuale, allora può essere possibile per un avversario riuscire a indovinarle e quindi compromettere completamente la sicurezza del sistema.

Negli anni '80, la crittografia moderna è stata posta su basi rigorose, e parte di questo processo è stato lo sviluppo di definizioni rigorose del concetto di sicurezza nella cifratura, nella firma digitale e nelle altre applicazioni di base. Un breakthrough in questa direzione fu il lavoro di Goldwasser e Micali [5] che introdusse la definizione moderna di sicurezza nella cifratura, da loro chiamata «sicurezza semantica» (semantic security). Senza entrare nei dettagli della definizione, essa garantisce che se un intruso ottiene accesso ad una comunicazione cifrata, ma non ne conosce le chiavi di cifratura, allora tale intruso non può estrarre da tale accesso *alcuna informazione sul contenuto dei messaggi del la quale non fosse già a conoscenza*. La definizione è basata su un esperimento mentale nel quale si confronta l'informazione che un intruso può ottenere monitorando il flusso di messaggi cifrati con l'informazione che può essere ottenuta senza praticare tale intrusione. Questo approccio definizionale è poi stato adattato nei successivi quarant'anni alla formalizzazione di molti altri problemi di critttografia ed è alla base del modo moderno di concettualizzare la sicurezza crittografica.

Un aspetto interessante di questi sviluppi è che, affinché un sistema di cifratura possa ottenere sicurezza semantica, è necessario che la stessa cifratura sia un processo probabilistico, nel quale della randomness «fresca», indipendente da tutte le scelte probabilistiche precedenti e successive, venga usata ogni volta che un nuovo messaggio viene cifrato.

Dato che i calcolatori elettronici sono delle macchine deterministiche, la necessità di scelte casuali nella generazione di chiavi crittografiche, e poi successivamente nella cifratura (così come nelle firme digitali e in altre applicazioni crittografiche) richiede dei metodi per la generazione di «bit random» da usare in tali applicazioni. Tale generazione casuale avviene osservando dei fenomeni fisici impredicibili, per esempio il rumore termico di un diodo, e poi applicando delle trasformazioni per convertire tali letture in una sequenza ideale di random bit, cioè una sequenza di bit in cui ciascun bit ha una probabilità del 50% di essere zero e del 50% di essere uno, e tali che i bit della sequenza siano mutuamente indipendenti statisticamente. Questa fase di trasformazione, chiamata *randomness extraction*, è di grande importanza per la sicurezza dei sistemi che lavorano a valle della generazione di numeri e bit casuali, ed è un'area di sviluppo nella quale c'è ancora molta distanza tra le soluzioni usate in pratica e i metodi che hanno una solida base teorica. Un possibile sviluppo futuro è l'uso di calcolatori quantistici nella produzione di sequenze di random bit per le quali è possibile offrire una garanzia di randomness e impredicibilità.

4. Generatori pseudocasuali e de-randomizzazione

Uno degli sviluppi della crittografia moderna è stata una definizione rigorosa del concetto di *generatore pseudocasuale*. Un generatore pseudocasuale è una procedura deterministica che viene inizializzata con un «seed», che è una breve sequenza di bit casuali generati nel modo descritto sopra, e, successivamente, il generatore produce una sequenza di bit «pseudocasuali» molto più lunga del seed, ma tale da essere *indistinguibile* da una sequenza completamente casuale usando test statistici efficienti. Senza entrare nei dettagli della definizione precisa di indistinguibilità (della quale c'è un ottimo trattamento nel Capitolo 3 della classica monografia di Goldreich [4] e che è legata alla nozione di sicurezza semantica) essa fa si che in qualunque applicazione computazionale sia necessaria una sequenza di bit completamente casuali, essa può essere sostituita dall'output di un generatore pseudocasuale, e quindi la quantità di random bit necessaria all'applicazione si riduce al numero di random bit della seed del generatore.

Un generatore pseudocasuale può quindi essere usato per ridurre il fabbisogno di random bit di un qualunque algoritmo probabilistico. Se tale numero si può ridurre sufficientemente, diventa allora possibile una simulazione completamente deterministica, semplicemente enu-

merando deterministicamente tutte le possibili seed del generatore.

Nisan e Wigderson [13] formularono nel 1988 una congettura di complessità computazionale e mostrarono che tale congettura implica l'esistenza di generatori pseudocasuali la cui seed è così breve che tale processo di enumerazione può essere eseguito per ogni algoritmo probabilistico efficiente, implicando una simulazione deterministica quasi altrettanto efficiente. Non era chiaro, però, quanto fosse plausibile la congettura sulla quale il lavoro di Nisan e Wigderson si basava. Nel decennio successivo, si riuscì a ricondurre tale congettura ad altri problemi aperti, e, culminando una lunga sequenza di altri risultati, Impagliazzo e Wigderson [7] mostrarono che la congettura di Nisan e Wigderson è equivalente ad una delle congetture standard di complessità computazionale.

Questa sequenza di lavori fu un punto di svolta nella comprensione degli algoritmi probabilistici, mostrando che, sotto congetture standard, tutti gli algoritmi probabilistici efficienti ammettono simulazioni puramente deterministiche che sono quasi altrettanto efficienti.

L'autore [16] successivamente dimostrò che il risultato principale di Impagliazzo and Wigderson può essere riformulato come un risultato più astratto e generale che dà anche una nuova soluzione al problema di *randomness extraction* discusso in precedenza. Questa connesione fu inaspettata, perché randomness extraction e generazione pseudocasuale sono due problemi che appartengono ad aree che si ritenevano ben distinte, il primo basato su teoria dell'informazione e statistica, il secondo sulla complessità computazionale. Questo nuovo approccio fu poi semplificato e ottimizzato in [16] e in lavori successivi, portando a nuove costruzioni di randomness extractors che hanno varie proprietà desiderabili, compresa l'affidabilità in contesti di comunicazione quantistica [3]. Queste idee hanno avuto una forte ricaduta sulla ricerca su randomness extraction nel decennio successivo.

Il problema della randomness extraction è studiato rispetto a vari modelli possibili per la sorgente di rumore dalla quale si vuole estrarre della randomness. Nell'importante modello di «due sorgenti indipendenti», un risultato recente di Chattopadhyay e Zuckerman [2], basato su nuove tecniche diverse da quelle descritte in precedenza, ha portato alla costruzione di un randomness extractor che, tra le altre applicazioni, risolve un problema aperto di Erdős in matematica pura che risale agli anni '50. Rimane ancora un problema aperto come trasferire tutte le nuove tecniche teoriche di randomness extraction degli ultimi vent'anni ai sistemi di produzione della randomness usati in pratica.

REFERENCES

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 2(160):781-793, 2004.

[2] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 670-683. ACM, 2016.

[3] Anindya De, Christopher Portmann, Thomas Vidick, and Renato Renner. Trevisan's extractor in the presence of quantum side information. *SIAM Journal on Computing*, 41(4): 915-940, 2012.

[4] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[5] Shafira Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270-299, 1984.

[6] W. K. Hastings. Monte Carlo sampling methods using Markov Chains and their applications. *Biometrika*, 57(1), 1970.

[7] Russell Impagliazzo and Avi Wigderson. P= BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 220-229, 1997.

[8] Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. MIP* = RE. *Communications of the ACM*, 64(11):131-138, 2021.

[9] Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison–Singer problem. *Annals of Mathematics*, pages 327-350, 2015.

[10] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6), 1953.

[11] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300-317, 1976.

[12] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[13] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149-167, 1994.

[14] Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128-138, 1980.

[15] Robert Solovay and Volker Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6(1):84-85, 1977.

[16] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860-879, 2001.