Catene di Markov

Hitting Time, Commute Time & Cover Time

Hitting Time:

Facciamo una passeggiata aleatoria su un grafo connesso, semplice e non orientato, partendo da un vertice *x* scegliamo random uno dei *d(x)* vertici, equiprobabili, su cui spostarci.

Definiamo l'hitting time come il tempo che impiega la catena di Markov per visitare y partendo da x.

$$H_{xy} = \min \{t>0 : X_t=y \}$$

E chiamiamo il tempo di primo ritorno, il tempo necessario per tornare a x dopo aver visitato y.

Teorema:

Sia G un grafo connesso e x e y suoi vertici. La probabilità che la passeggiata aleatoria su G raggiunga y prima di tornare a x è data da:

$$P_x = \frac{1}{d(x)\, r_{xy}}$$

Dove con r_{xy} indichiamo la resistenza effettiva tra i vertici x e y in una rete.

Se scegliamo un vertice y del grafo randomicamente secondo la distribuzione π, il tempo atteso per visitare il vertice y partendo da uno specifico vertice iniziale a non dipende da a. Questo tempo si chiama target time:

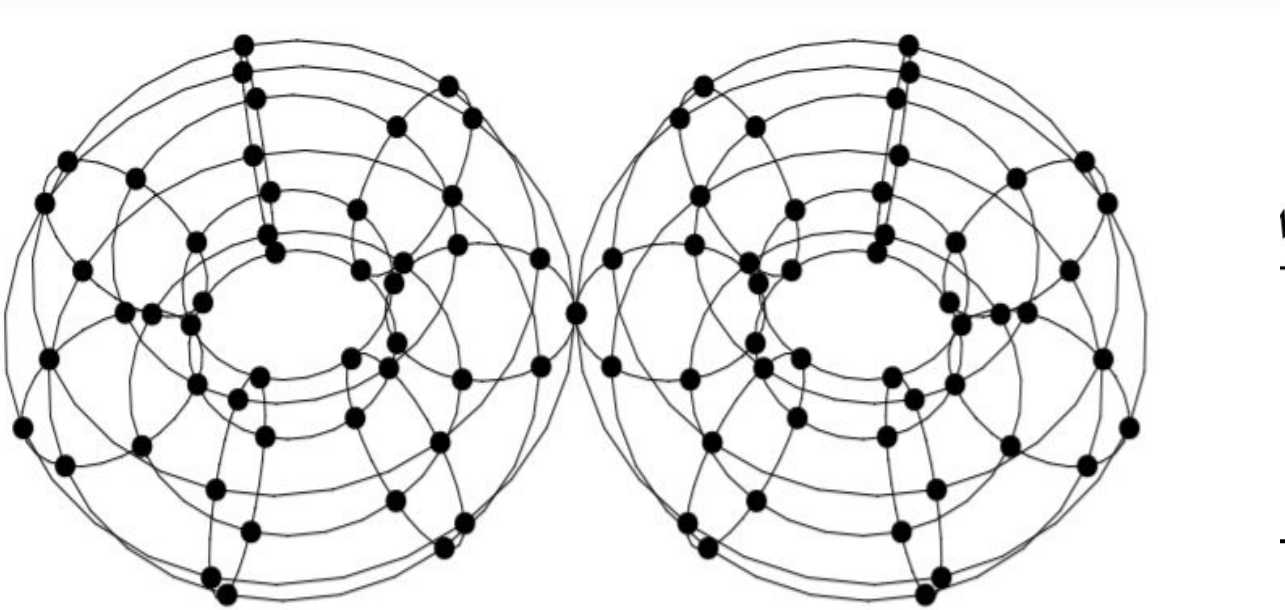
$$t_{\odot}^{a} := \sum \mathbf{E}_a(\tau_x)\pi(x)$$

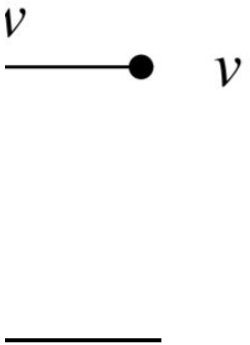
Poiché questo tempo non dipende da a, allora è possibile definire il target time nel modo seguente:

$$t_{\odot} = \sum \pi(x)\pi(y)\mathbf{E}_x(\tau_y) = \mathbf{E}_{\pi}(\tau_{\pi})$$

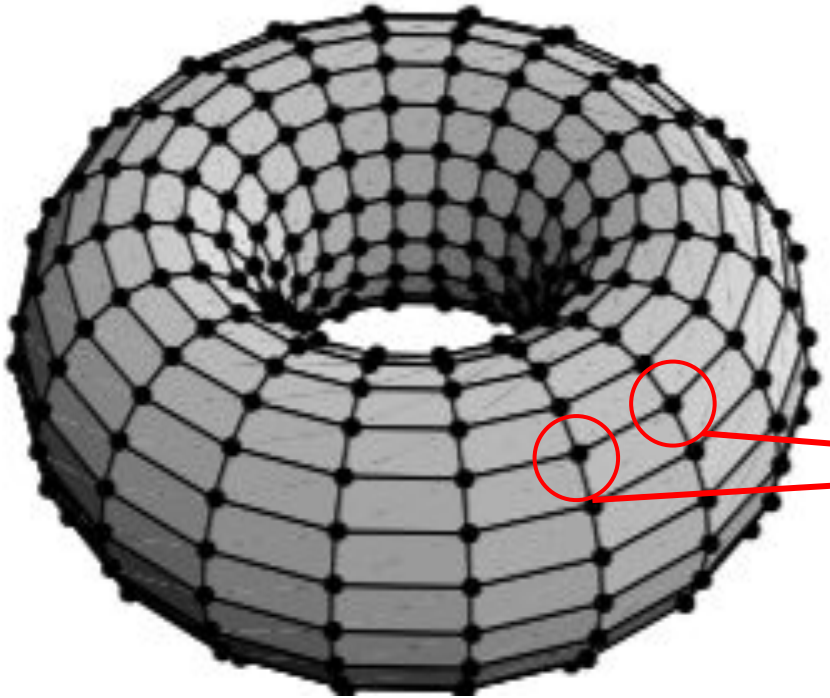
Il target time è utile per stimare l'hitting time nel caso peggiore, definito come segue:

$$t_{\text{hit}} := \max_{x,y \in \mathcal{X}} \mathbf{E}_x(\tau_y)$$





Hitting time su un toro:

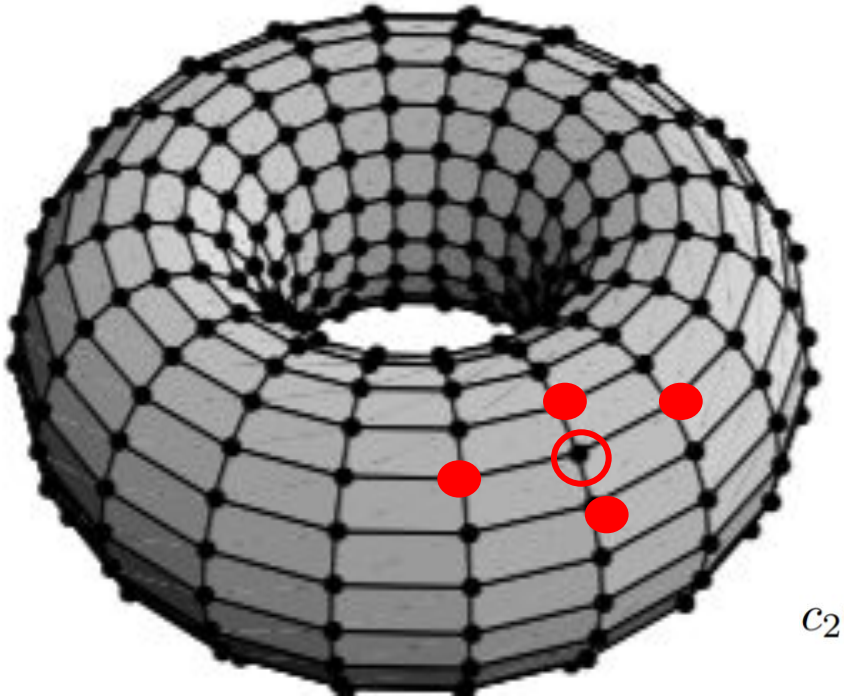


Il toro d-dimensionale è un grafo in cui l'insieme dei vertici coincide con il prodotto cartesiano:

$$\mathbb{Z}_n^d = \underbrace{\mathbb{Z}_n \times \cdots \times \mathbb{Z}_n}_{\textit{d-volte}}.$$

$$\boldsymbol{x} = (x^1, \ldots, x^d)$$
$$\boldsymbol{y} = (y^1, y^2, \ldots, y^d)$$

$$x^i = y^i \qquad i \neq j \qquad j \in \{1, 2, \ldots, d\}$$
$$x^j \equiv y^j \pm 1 \mod n.$$



Se n è pari allora il grafo è bipartito e la camminata aleatoria sul grafo è periodica. Quindi considereremo una passeggiata aleatoria lazy.

Nella figura abbiamo considerato n=20 e d=2.

$$\mathbf{E}_a(\tau_b) = \mathbf{E}_b(\tau_a)$$

$$c_d n^d \leq \mathbf{E}_x(\tau_y) \leq C_d n^d \qquad d \geq 3$$

$$c_2 n^2 \log(k) \leq \mathbf{E}_x(\tau_y) \leq C_2 n^2 \log(k+1) \qquad d = 2$$

Hitting time su un albero:

Sia T un albero finito pesato, quindi ogni spigolo (x,y) ha peso c(x,y)>0.
Rimuoviamo tutti gli spigoli contenenti y, quindi T diventerà un grafo sconnesso.
Consideriamo solo la componente connessa con x, e a questa aggiungiamo lo spigolo {x,y}. Questo nuovo grafo lo chiamiamo $\tilde{T}$.

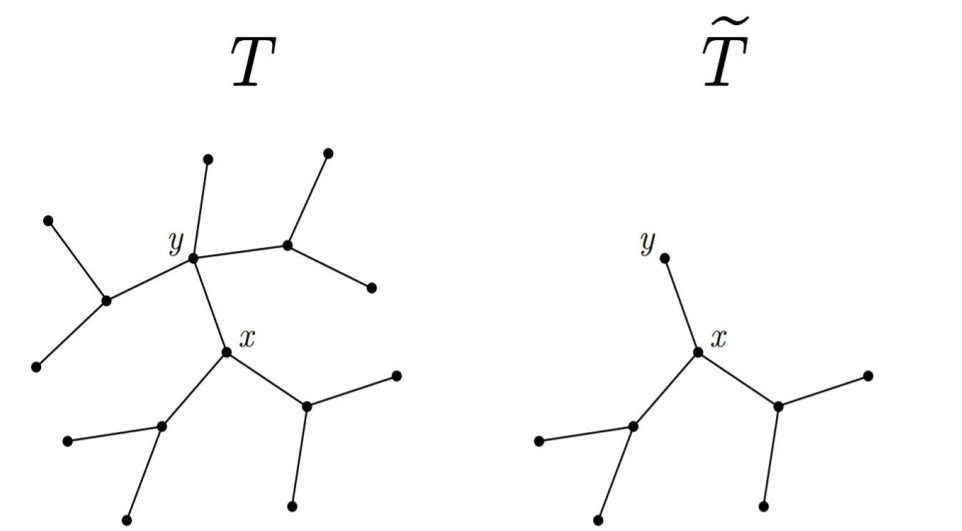
$$\tilde{c} = \sum_{u,v} \tilde{c}(u,v)$$

$$\tilde{\mathbf{E}}_y[\tau_y^+] = \frac{1}{\tilde{\pi}(y)} = \frac{\tilde{c}}{\tilde{c}(x,y)}$$

$$\tilde{\mathbf{E}}_y(\tau_y^+) = 1 + \mathbf{E}_x(\tau_y)$$

o di ritorno a y deve essere uguale a 1 più

$$\mathbf{E}_x(\tau_y) = \begin{cases} 2|\tilde{E}| - 1 \\ \frac{\tilde{c}}{\tilde{c}(x,y)} - 1 \end{cases}$$



Commute Time:

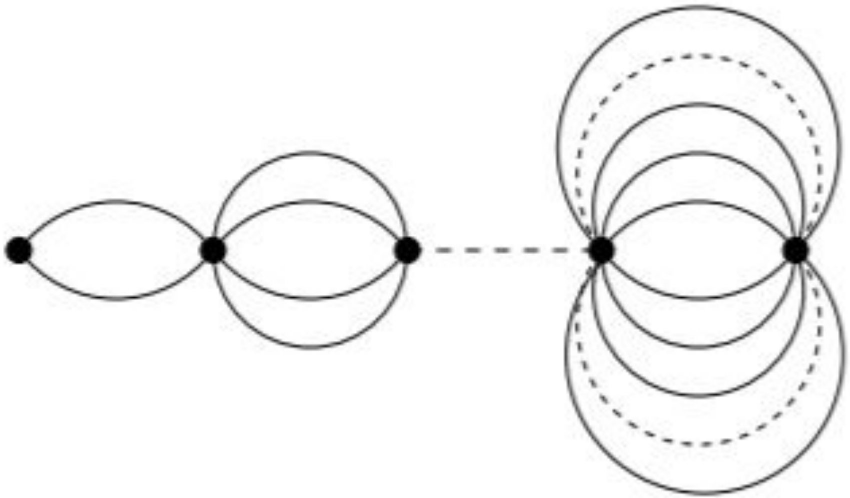
Il commute time è il tempo necessario ad andare da x a y e poi tornare indietro a x. Perciò possiamo scriverlo nel modo seguente:

$$C_{xy} := H_{xy} + H_{yx}$$

Teorema: Siano x e y vertici distinti in G e |E|=m. Allora:

$$C_{xy} = 2m\mathrm{r}_{xy}$$

Commute Time su un albero binario:



Consideriamo un albero binario finito con radice r e di profondità k. Sia n il numero di vertici dell'albero allora il numero di spigoli sarà n-1.

Vogliamo calcolare il commute time tra la radice e l'insime delle foglie B. Identifichiamo tutti i vertici al livello j=1,...,k.

Ciò che otteniamo è equivalente a un segmento di lunghezza k, dove tra il vertice j-1 e il vertice j ci sono 2^j lati.

Otteniamo che la resistenza tra la radice e l'insieme di foglie B è:

$r_{rB} = 1-2^{-k} < 1$

Quindi $C_{rB} = 2(n-1)r_{rB}$

$$\mathbf{E}_{x,y}(\tau) = k(n-k)$$ata

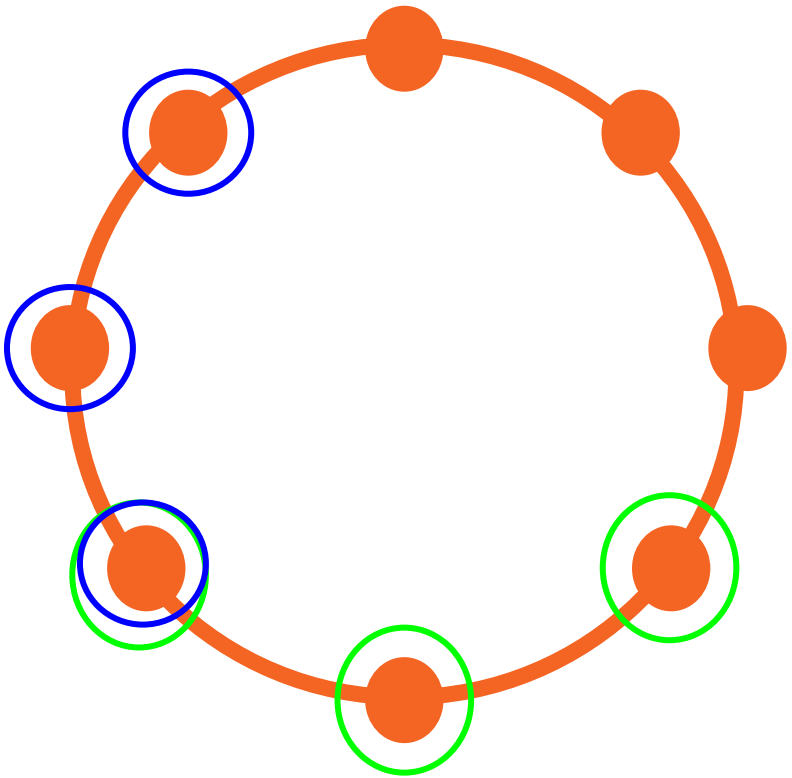
Dove k è la distanza (in senso orario) tra x e y.

Voglio costruire un coupling (X_t , Y_t).
Scegliamo due vertici di partenza x e y,

$$t_{\text{hit}} = \max_{x,y} \mathbf{E}_x(\tau_y) = \left\lfloor \frac{n^2}{2} \right\rfloor$$ ne · una

...neta e implementiamo la passeggiata aleatoria lazy.

Chiamiar $t_{\text{mix}} \leq n^2$nza (in senso orario) tra X e Y.



Cover Time:

Il tempo di ricoprimento è il tempo necessario alla catena a visitare ogni vertice del grafo.

Esempio: Sia c_n il tempo necessario a visitare n distinti vertici.
Supponiamo che abbiamo appena visitato il vertice (n-1)-esimo.
Sappiamo che il tempo necessario a visitare l'n-esimo vertice è n-1 (rovina del giocatore).

Ne segue che: $c_n = c_{n-1} + (n-1)$

iterando si ha: $c_n = n(n-1)/2$

Cover Time:

Teorema: Il tempo di ricoprimento su un grafo G è al massimo 2m(n-1).

Consideriamo l'albero generante T di G. Supponiamo di fare un passeggiata ($v=v_0,\dots,v_{2n-2}=v$).

Otterremo:

$$= \sum_{xy \in E(T)} C_{xy} = 2m \sum_{xy \in E(T)} r_{xy} = \frac{2m}{t(G)} \sum_{xy \in E(T)} t_{xy}(G)$$

Vogliamo trovare delle stime per il cover time in relazione al hitting time.
Ovviamente se ogni passeggiata inizia dal vertice x, allora il tempo per visitare y sarà minore o uguale del tempo per visitare tutti i nodi del grafo.

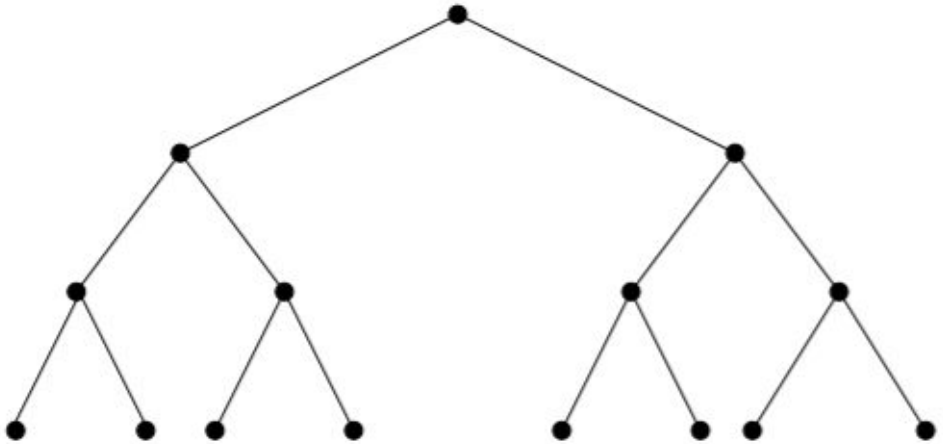
$$t_{\text{hit}} = \mathbf{E}_x \tau_y \leq \mathbf{E}_x \tau_{\text{cov}} \leq t_{\text{cov}}$$

Grazie al metodo di Metthewes possiamo ottenere delle stime migliori:

$$t_{\text{cov}} \leq t_{\text{hit}} \left(1 + \frac{1}{2} + \cdots + \frac{1}{n-1}\right)$$

$$t_{\text{cov}} \geq \max_{A \subseteq \mathcal{X}} t^A_{\text{min}} \left(1 + \frac{1}{2} + \cdots + \frac{1}{|A|-1}\right) \qquad t^A_{\text{min}} = \min_{a,b \in A, a \neq b} \mathbf{E}_a(\tau_b)$$

Cover Time su un albero binario:



Consideriamo un albero binario finito con radice r e di profondità k. Avremo $n=2^{k+1}-1$ vertici. L'hitting time massimale si otterrà considerando due radici il cui vertice in comune più vicino è r.

In questo caso l'hitting time coincide con il commute time tra r e una delle foglie.

Poichè la resistenza effettiva tra le foglie è k, otteremo:

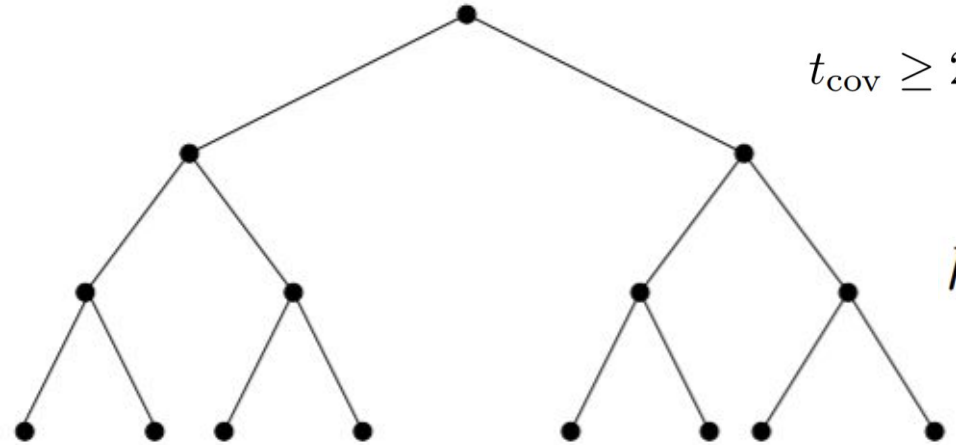
$$\mathbf{E}_a \tau_b = 2(n-1)k$$

$$t_{\text{cov}} \leq 2(n-1)k\left(1+\frac{1}{2}+\cdots+\frac{1}{n}\right) = (2+o(1))(\log 2)nk^2$$

Per ottenere la stima dal basso dobbiamo considerare un opportuno sottoinsieme A di G(V). Fissiamo un livello h dell'albero e sia A l'insieme delle 2^h foglie tali che ogni vertice al livello h ha un unico discendente in A. Dobbiamo scegliere un valore di h tale che la stima sia ottimale.
Consideriamo le foglie a,b, h'<h è il livello a cui le due foglie hanno il loro primo antenato comune. Otteniamo:

$$\mathbf{E}_a \tau_b = 2(n-1)(k-h')$$

minimizzato quando $h' = h - 1$



$$t_{\mathrm{cov}} \geq 2(n-1)(k-h+1)\left(1 + \frac{1}{2} + \cdots + \frac{1}{2^h - 1}\right)$$

$$h = \lfloor k/2 \rfloor \qquad t_{\mathrm{cov}} \geq \frac{1}{4} \cdot (2 + o(1))(\log 2) n k^2$$

Cover time e alberi di ricoprimento:

La ricerca in profondità (DFS) per un albero T può aiutarci a stimare il cover time per un grafo qualsiasi G. Possiamo definire la DFS di T, con profondità n>1, come il seguente cammino: $v_0\Gamma_1 v_0\Gamma_2 ... \Gamma_m v_0$.

Teorema: Sia T un albero ricoprente di G. Il cover time di una passeggiata aleatoria su un grafo G soddisfa la seguente:

$$t_{\text{cov}} \leq 2|E| \sum_{(x,y)\in T} \mathcal{R}(x \leftrightarrow y) \leq 2(n-1)|E|$$

$$t_{\text{cov}} \leq \sum_{i=1}^{2n-2} \mathbf{E}_{x_{i-1}} \tau_{x_i} = 2 \sum_{(x,y)\in T} \mathcal{R}(x \leftrightarrow y)|E|$$

Utilizzando questo teorema possiamo dare una stima del cover time per un grafo d-regolare.

Teorema: Per una passeggiata aleatoria su un grafo d-regolare con n vertici il cover time soddisfa:

$$t_{\text{cov}} \leq 3n^2$$