Breve introduzione alla NP-completezza

1 Problemi decisionali

Un problema decisionale è un problema che ammette una risposta di tipo booleano - sì oppure no.

Esempio 1 - Circuito Hamiltoniano (HC): dato un grafo $G = (V, E)$ esiste un ciclo in G che passa una e una sola volta per ciascun nodo in V?

Esempio 2 - Short Path (SP): dati un grafo $G = (V, E)$, una coppia di nodi $u, v \in V$ e un intero k, esiste in G un percorso da u a v di lunghezza $\leq k$?

Esempio 3 - Satisfiability (SAT): dati un insieme X di variabili booleane e una funzione $f : \{\text{vero}, \text{falso}\}^X \to \{\text{vero}, \text{falso}\}$, esiste una assegnazione di verità alle variabili in X $\alpha : X \to \{\text{vero}, \text{falso}\}$ tale che $f(\alpha(X)) = \text{vero}$?

2 Istanze di problemi decisionali

L'insieme delle istanze di un problema decisionale è la descrizione dell'insieme dei dati di quel problema.

Esempio 1 - Circuito Hamiltoniano (HC): dato un grafo $G = (V, E)$, esiste un ciclo in G che passa una e una sola volta per ciascun nodo in V?

L'insieme delle istanze di HC è $I_{HC} = \{< G = (V, E) >: G \text{ è un grafo non orientato }\}$.

Esempio 2 - Short Path (SP): dati un grafo $G = (V, E)$, una coppia di nodi $u, v \in V$ e un intero k, esiste in G un percorso da u a v di lunghezza $\leq k$?

L'insieme delle istanze di SP è $I_{SP} = \{< G = (V, E), u, v, k >: G \text{ è un grafo } \wedge u, v \in V \wedge k \in \mathbb{N}\}$.

Esempio 3 - Satisfiability (SAT): dati un insieme X di variabili booleane e una funzione $f : \{\text{vero}, \text{falso}\}^X \to \{\text{vero}, \text{falso}\}$, esiste una assegnazione di verità alle variabili in X $\alpha : X \to \{\text{vero}, \text{falso}\}$ tale che $f(\alpha(X)) = \text{vero}$?

L'insieme delle istanze di SAT è $I_{SAT} = \{< X, f >: f : \{\text{vero}, \text{falso}\}^X \to \{\text{vero}, \text{falso}\}\}$.

Un'istanza di un problema decisionale Γ è un elemento di I_Γ.

3 Problemi decisionali in P

Consideriamo il problema SP e il seguente algoritmo che lo decide.

Algoritmo A-SP

Input: $G = (V, E)$, $u, v \in V$, $k \in \mathbb{N}$

Fase 1: mediante l'algoritmo di Dijkstra (ad esempio), calcola il percorso p di lunghezza minima fra u e v (se un tale percorso esiste).

Fase 2: verifica se la lunghezza di p è $\leq k$; in caso affermativo rispondi sì, altrimenti rispondi no.

Il numero di "passi" eseguiti da A-SP è proporzionale a $|V|^2$ - ossia è $O(|V|^2)$ e quindi il problema SP appartiene alla classe P.

La classe P è la classe dei problemi di decisione che sono decisi da un algoritmo che opera in un numero di "passi" polinomiale nella dimensione dell'istanza.

OSSERVAZIONE: l'algoritmo A-SP è di tipo costruttivo, ossia, per decidere se esiste in G un percorso fra u e v di lunghezza $\leq k$ prova a costruire un siffatto percorso. Comunque, per rispondere al quesito di un problema di decisione l'algoritmo potrebbe seguire strade diverse da quella costruttiva - ad esempio, potrebbe dimostrare se qualche proprietà connessa al quesito è vera.

Ad esempio, per decidere se un grafo è planare è sufficiente verificare se esso soddisfa la formula di Eulero, senza dover descriverne un disegno sul piano.

4 Problemi decisionali in NP

Consideriamo il problema HC: per questo problema non è stato fino ad ora possibile progettare un algoritmo che lo decida in tempo polinomiale. Ossia, utilizzando un numero di "passi" proporzionale alla dimensione della sua istanza, dove la dimensione dell'istanza $< G = (V, E) >$ è, sostanzialmente, il numero di bit necessari a descrivere G.

Però, potremmo chiedere consiglio ai nostri amici. Potremmo, cioè, pensare ad un algoritmo della forma seguente:

Algoritmo A1-HC

Input: $G = (V, E)$

Fase 1: chiedi ad un amico bravo (diciamo, un genio) se G contiene un ciclo che passa una e una sola volta per ciascun nodo.

Fase 2: se l'amico dice di sì allora la risposta è sì, altrimenti la risposta è no.

Bello, intuitivo, facile, ma questo ragionamento ha una pecca... Come faccio a fidarmi della risposta del mio amico (seppur bravo)?

Eh, no! Non posso fidarmi della risposta del mio amico (seppur bravo). Devo almeno chiedergli una prova che quel che dice è vero! Una prova della quale non mi fiderò, ma che andrò a verificare!

Modifichiamo, allora, l'algoritmo A1-HC nel modo seguente:

Algoritmo NA-HC

Input: $G = (V, E)$

Fase 1: chiedi ad un amico bravo (un genio!) un ciclo c che passa una e una sola volta per ciascun nodo di G, se un tale ciclo esiste.

Fase 2: verifica se c passa davvero una e una sola volta per ciascun nodo di G e, in tal caso, la risposta è sì.

Ora, verificare 'se c passa davvero una e una sola volta per ciascun nodo di G" richiede tempo polinomiale nella dimensione di G. Perciò, se disponessimo di un genio capace di suggerirci il ciclo c giusto, in tempo polinomiale riusciremmo a rispondere sì correttamente.

OSSERVAZIONE 1: esattamente come nel caso dei problemi in P, non è detto che il genio ci debba suggerire una prova costruttiva del fatto che la nostra istanza ha risposta sì. Potrebbe fornirci, genericamente, una dimostrazione del fatto che la nostra istanza ha risposta sì. La cosa importante è che per ogni istanza che ha risposta sì il genio può fornirci una prova di questo fatto e che tale prova può essere verificata in tempo polinomiale nella dimensione dell'input.

OSSERVAZIONE 2: 'per ogni istanza che ha risposta sì il genio può fornirci una prova di questo fatto e tale prova può essere verificata in tempo polinomiale nella dimensione dell'input'. Bene! Ma se l'istanza ha risposta no?!

Riflettiamo: se il nostro genio ci comunica un ciclo che non passa per tutti i nodi di G, o che passa più volte per lo stesso nodo, cosa possiamo concludere?

Riflettiamo: per poter concludere che G non contiene un ciclo hamiltoniano è necessario che nessun ciclo in G passi una e una sola volta per ciascun nodo; non è certo sufficiente il singolo ciclo c mostratoci dal genio!

Perciò, possiamo concludere soltanto che il nostro genio non ha saputo trovare un ciclo hamiltoniano in G, ma non sappiamo se non l'ha trovato perchè non esiste o perche lui non è abbastanza geniale!

5 La classe NP

- La classe NP è la classe dei problemi verificabili in tempo polinomiale.
- Ossia, un problema è in NP se:
  - esiste un genio (del quale non mi fido) tale che
  - per ogni istanza del problema che ha risposta sì

  - il genio mi sa suggerire una prova che quell'istanza ha risposta sì
  - e quella prova io posso verificarla in tempo polinomiale nella dimensione dell'istanza.

- Ma, in definitiva, il genio possiamo evitare di tirarlo in ballo:
- Ossia, un problema è in NP se:
  - per ogni istanza del problema che ha risposta sì
  - esiste una prova che quell'istanza ha risposta sì
  - e quella prova io posso verificarla in tempo polinomiale nella dimensione dell'istanza.

6 P è contenuto in NP

Ri-pensiamo al problema SP. L'algoritmo A-SP è stato progettato proprio per decidere SP.

Dunque, se l'esecuzione dell'algoritmo A-SP con input una qualche istanza $< G = (V, E), u, v, k >$ risponde sì, è detta esecuzione stessa una prova che $< G = (V, E), u, v, k >$ ha risposta sì.

Perciò, verificare la prova significa, sostanzialmente,... ripetere l'esecuzione di A-SP con input $< G = (V, E), u, v, k >$.

Ossia, in qualche modo, prova e verifica coincidono.

E poiché l'esecuzione di A-SP con input $< G = (V, E), u, v, k >$ termina entro un numero di "passi" polinomiale nella dimensione di $< G = (V, E), u, v, k >$, questo prova che SP $\in$ NP!

E siccome lo stesso ragionamento lo possiamo ripetere per ogni problema appartenente a P, possiamo concludere che:

$$P \subseteq NP$$

7 La congettura fondamentale

- Dunque, $P \subseteq NP$.
- Ma, sino ad ora non si è riusciti a dimostrare se si tratta di una relazione di inclusione stretta oppure di una uguaglianza.
- Anche se si congettura che sia $P \neq NP$.
- E siccome NP contiene moltissimi problemi di notevole rilevanza applicativa che non si riesce a collocare in P, sulla soluzione della congettura $P \neq NP$ – che è la congettura fondamentale della teoria della complessità computazionale – è stato posto un premio di un milione di dollari.

- Per provare a individuare i problemi separatori fra P e NP, ossia, i problemi appartenenti a $NP - P$, sono stati introdotti i concetti di riduzione polinomiale e di NP-completezza.

8 Riduzioni polinomiali

Un problema Γ è riducibile polinomialmente a un problema Δ se esiste un algoritmo A che, presa in input una istanza x di Γ, in un numero di "passi" polinomiale nella lunghezza di x calcola una istanza y di Δ in modo tale che:

$$x \text{ ha risposta sì (per } \Gamma) \text{ se e soltanto se } y \text{ ha risposta sì (per } \Delta).$$

Se Γ è riducibile polinomialmente a Δ scriviamo $\Gamma \leq \Delta$.

In pratica: se so ridurre polinomialmente Γ a un problema Δ che so decidere, allora posso combinare la riduzione con l'algoritmo che decide Δ ed ottenere un algoritmo che decide Γ e non impiega "troppi più passi" di quelli impiegati per decidere Δ.

9 P e la riducibilità polinomiale

Se so ridurre polinomialmente Γ a un problema Δ che so decidere allora posso combinare la riduzione con l'algoritmo che decide Δ ed ottenere un algoritmo che decide Γ e non impiega "troppi più passi" di quelli impiegati per decidere Δ.

In particolare, vale il seguente:

Teorema 1 (Chiusura di P rispetto alla riducibilità polinomiale)**.** *Siano* Γ *e* Δ *due problemi decisionali; se* $\Gamma \leq \Delta$ *e* $\Delta \in P$ *allora* $\Gamma \in P$*.*

10 Il problema SAT

Consideriamo il seguente problema decisionale:

Satisfiability (SAT): dati un insieme X di variabili booleane e una funzione $f : \{\text{vero}, \text{falso}\}^X \to \{\text{vero}, \text{falso}\}$, esiste una assegnazione di verità alle variabili in X $\alpha : X \to \{\text{vero}, \text{falso}\}$ tale che $f(\alpha(X)) = \text{vero}$?

Nel 1971 è stato dimostrato il seguente:

Teorema 2 (di Cook-Levin)**.** *Per ogni problema di decisione* $\Gamma \in NP$ *vale che* $\Gamma \leq SAT$*.*

Il teorema di Cook-Levin e il teorema di chiusura di P rispetto alla riducibilità polinomiale mostrano che:

Se esistesse un algoritmo polinomiale per decidere SAT allora, per ogni $\Gamma \in NP$ esisterebbe un algoritmo polinomiale per decidere Γ.

Ossia,

Se esistesse un algoritmo polinomiale per decidere SAT allora sarebbe $P = NP$.

SAT $\in$ NP: infatti se $< X, f >$ è un'istanza sì di SAT allora una prova di ciò è una assegnazione di verità alle variabili in X (che può essere verificata in tempo proporzionale alla dimensioni di f).

Inoltre, come abbiamo visto:

Se esistesse un algoritmo polinomiale per decidere SAT allora sarebbe $P = NP$.

E, dunque, possiamo affermare che:

Se $P \neq NP$ allora $SAT \in NP - P$.

SAT è un possibile problema separatore fra P e NP.

11 I problemi NP-completi

Esistono in NP numerosi problemi con le stesse caratteristiche di SAT: sono i problemi NP-completi.

Un problema Δ è NP-completo se:

1. $\Delta \in NP$
2. per ogni problema $\Gamma \in NP$ vale che $\Gamma \leq \Delta$

Dunque, SAT è NP-completo.

Dunque, come SAT, i problemi NP-completi sono i possibili separatori fra P e NP.

Per dimostrare che un problema $\Delta \in NP$ è NP-completo si usa il seguente:

Teorema 3. *Se Γ è NP-completo e $\Gamma \leq \Delta$ (con $\Delta \in NP$), allora Δ è NP-completo.*

Ossia, per dimostrare che un problema $\Delta \in NP$ è NP-completo è sufficiente scegliere un problema Γ che già sappiamo essere NP-completo e mostrare che $\Gamma \leq \Delta$.

E di problemi NP-completi noti ce ne sono a bizzeffe: HC, CLIQUE, VERTEX COVER, DOMINATING SET, TRAVELLING SALESMAN PROBLEM, ... solo per citarne alcuni su grafi.