Problemi di ottimizzazione

Un problema di ottimizzazione è formato da un insieme di istanze, dove ogni istanza è rappresentata da una coppia (f, S) con

$$f : S \to R$$

detta *funzione obiettivo* e S detta *regione ammissibile*.

Problemi di massimo e minimo

Se il problema è di massimo, allora l'istanza si rappresenta nel modo seguente

$$\max_{x \in S} f(x)$$

e risolvere l'istanza vuol dire trovare un $x^* \in S$ tale che

$$f(x^*) \geq f(x) \quad \forall\, x \in S.$$

Se il problema è di minimo, allora l'istanza si rappresenta nel modo seguente

$$\min_{x \in S} f(x)$$

e risolvere l'istanza vuol dire trovare un $x^* \in S$ tale che

$$f(x^*) \leq f(x) \quad \forall\, x \in S.$$

In entrambi i casi il punto x^* viene chiamato soluzione ottima dell'istanza, mentre $f(x^*)$ viene detto valore ottimo dell'istanza.

Classificazione

I problemi di ottimizzazione possono essere raggruppati in due grandi categorie.

- **Ottimizzazione Combinatoria** In ogni istanza la regione ammissibile S contiene un numero finito o un'infinità numerabile di punti.
- **Ottimizzazione Continua** La regione ammissibile S può contenere un'infinità non numerabile di punti.

SHORTEST PATH

Nei problemi SHORTEST PATH (cammino a costo minimo) dato un grafo orientato $G = (V, A)$ con costo (distanza) d_{ij} per ogni $(i, j) \in A$ e dati due nodi $s, t \in V$, $s \neq t$, vogliamo individuare un cammino orientato da s a t di costo minimo.

MST

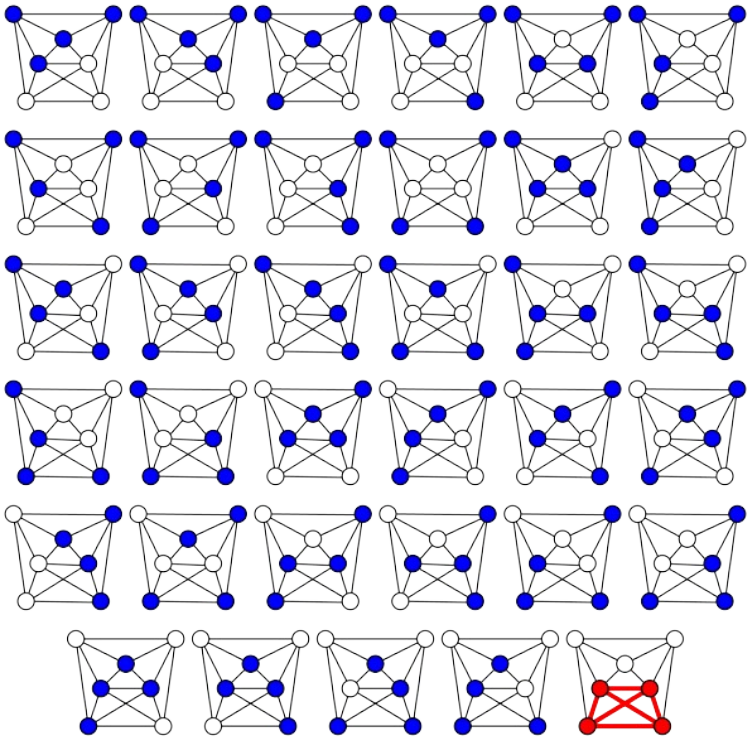
Nei problemi MST (Minimum Spanning Tree o albero di supporto a peso minimo), dato un grafo non orientato $G = (V, E)$ con pesi w_{ij} per ogni $(i, j) \in E$, si vuole determinare tra tutti gli alberi di supporto $T = (V, E_T)$ del grafo quello con peso complessivo $\sum_{(i,j) \in E_T} w_{ij}$ minimo.

CLIQUE

Dato un grafo $G = (V, E)$, vogliamo individuare un sottinsieme $C \subseteq V$ dei nodi tali che

$$\forall\, i, j \in C,\ i \neq j\ :\ (i, j) \in E$$

(il sottografo indotto da C è completo) di cardinalità $|\ C\ |$ massima.



TSP

Nel problema TSP (Travelling Salesaman Problem o problema del commesso viaggiatore), dato un grafo orientato completo $G = (V, A)$ con le relative distanze degli archi rappresentate dagli interi non negativi d_{ij}, per ogni $(i, j) \in A$, individuare nel grafo un circuito hamiltoniano (ciclo che tocca tutti i nodi del grafo una e una sola volta) di distanza totale (somma delle distanze dei suoi archi) minima.

In pratica, si tratta di scegliere una permutazione $i_1, \ldots, i_n$ degli $n = \mid V \mid$ nodi del grafo tale che il circuito

$$i_1 \rightarrow \cdots \rightarrow i_n \rightarrow i_1$$

sia di lunghezza minima.

Si noti che, fissando il nodo iniziale i_1, il numero di possibili circuiti hamiltoniani è pari a $(n-1)!$

Sottoclassi TSP

Tra le sottoclassi di TSP citiamo il TSP *simmetrico* ($d_{ij} = d_{ji}$ per ogni i, j), che a sua volta contiene la sottoclasse del TSP *metrico* dove in più le distanze soddisfano la diseguaglianza triangolare

$$d_{ij} \leq d_{ik} + d_{kj} \quad \forall\, i, j, k \text{ distinti.}$$

KNAPSACK

Nel problema KNAPSACK (o problema dello zaino), dati n oggetti aventi come pesi gli interi positivi $p_1, \ldots, p_n$ e come valori gli n interi positivi $v_1, \ldots, v_n$ e dato uno zaino con capacità pari a un intero positivo b, vogliamo individuare un sottinsieme K degli n oggetti con peso complessivo $\sum_{i \in K} p_i$ non superiore a b e valore complessivo $\sum_{i \in K} v_i$ massimo.

Difficoltà problemi di ottimizzazione

Dato un problema di ottimizzazione il nostro scopo è ovviamente risolverlo, restituendone una soluzione ottima e il valore ottimo.

Per risolverlo abbiamo bisogno di un algoritmo di risoluzione $\mathcal{A}$ per il problema, ovvero una procedura che, ricevuti in input i dati che definiscono un'istanza, restituisce in output proprio una soluzione ottima e il valore ottimo di quell'istanza.

Esistenza algoritmo

Se ci limitiamo ai problemi di ottimizzazione combinatoria in cui la regione ammissibile è costituita da un numero finito di elementi (vale per i problemi `MST`, `SHORTEST PATH`, `CLIQUE`, `TSP`, `KNAPSACK`), è abbastanza semplice identificare un algoritmo di risoluzione.

Infatti se la regione ammissibile S contiene un numero finito di elementi la seguente procedura, detta di enumerazione completa, risolve il problema: valutare la funzione f per ogni elemento in S e restituire l'elemento con valore di f minimo (o massimo se il problema è di massimo).

Tuttavia ...

... una semplice osservazione metterà in luce i limiti di questo approccio.

Consideriamo il problema TSP su un grafo con numero di nodi $n = 22$. Abbiamo osservato come in tale grafo il numero di circuiti hamiltoniani è pari a $(n-1)! = 21! > 10^{19}$.

Supponiamo (ottimisticamente) che la valutazione della funzione obiettivo per un singolo circuito hamiltoniano richieda un nanosecondo (10^{-9} secondi).

Ne consegue che la valutazione di tutti i circuiti hamiltoniani richiede più di 10^{10} secondi che corrispondono ad un tempo superiore ai 200 anni.

Continua

Ancora più impressionante è notare quello che succede se si aumenta di uno il numero di nodi: il tempo richiesto supera i 4000 anni!

Ciò dimostra che sebbene sia sempre possibile teoricamente risolvere tali problemi, in pratica dobbiamo fare i conti con dei limiti temporali e quindi, da un punto di vista pratico, si parlerà di problema risolvibile con una data procedura solo nel caso in cui la procedura restituisca una soluzione in tempi ragionevoli.

Ciò rende la procedura di enumerazione completa accettabile solo per istanze di problemi di dimensioni limitate (quindi, ad esempio, per grafi con pochi nodi per i problemi TSP).

Domanda

Esistono altre procedure che consentono di risolvere in tempi ragionevoli istanze dei problemi con dimensioni molto più elevate?

La risposta è: dipende dal problema.

È in generale vero che esistono procedure molto più efficienti dell'enumerazione completa ma mentre per alcuni (come il problema `MST`) si possono risolvere in tempi ragionevoli istanze di grandi dimensioni, per altri, come il problema `KNAPSACK` e, ancor più, per il problema `TSP` *può* essere difficile risolvere anche istanze di dimensioni non troppo elevate (si osservi il *può*: con algoritmi sofisticati si sono risolte anche istanze di `TSP` con più di 15000 nodi, ma gli stessi algoritmi possono essere messi in crisi da istanze più piccole).

Tutto ciò ha una formulazione ben precisa nella teoria della complessità, alla quale giungeremo dopo aver discusso di complessità degli algoritmi.

Complessità

Dimensione istanza $I \rightarrow dim(I)$ = quantità di memoria necessaria per memorizzare (in codifica binaria) l'istanza I.

Procedura di risoluzione o algoritmo $\mathcal{A}$ per il problema.

$numop_{\mathcal{A}}(I)$= numero di operazioni elementari eseguite da $\mathcal{A}$ per risolvere I.

Analisi worst case

L'analisi *worst case* definisce il tempo $t_{\mathcal{A}}(k)$ necessario all'algoritmo $\mathcal{A}$ per risolvere istanze di dimensione k come il *massimo* tra tutti i tempi di esecuzione di istanze di dimensione k, cioè

$$t_{\mathcal{A}}(k) = \max_{I:\ dim(I)=k} numop_{\mathcal{A}}(I).$$

Continua

Tipicamente non si conosce l'espressione analitica della funzione $t_{\mathcal{A}}(k)$ ma se ne conosce l'ordine di grandezza. Si dice che la funzione $t_{\mathcal{A}}(k) = O(g(k))$, ovvero che $t_{\mathcal{A}}(k)$ è dell'ordine di grandezza della funzione $g(k)$, se esiste una costante $u > 0$ tale che

$$t_{\mathcal{A}}(k) \leq ug(k).$$

Diverse possibili funzioni g

$g(k)$	$k = 10$	$k = 20$	$k = 30$	$k = 40$	$k = 50$
$\log_2(k)$	3.32	4.32	4.90	5.32	5.64
k	10	20	30	40	50
k^2	100	400	900	1600	2500
k^3	1000	8000	27000	64000	125000
2^k	1024	$> 10^6$	$> 10^9$	$> 10^{12}$	$> 10^{15}$
$k!$	3628800	$> 10^{18}$	$> 10^{32}$	$> 10^{47}$	$> 10^{64}$

Complessità degli algoritmi

Un algoritmo per il quale $t_{\mathcal{A}}(k)$ fosse dell'ordine di grandezza di 2^k oppure $k!$ si dice che ha *complessità esponenziale*. È evidente che un tale algoritmo consente di risolvere in tempi ragionevoli solo istanze di dimensioni limitate.

Per questa ragione si cercano per i problemi algoritmi di risoluzione con *complessità polinomiale*, in cui cioè la funzione $t_{\mathcal{A}}$ è dell'ordine di grandezza di un polinomio k^p per un qualche esponente p.

Continua

Tanto maggiore è l'esponente p del polinomio, quanto più rapidamente cresce il numero di operazioni dell'algoritmo al crescere di k e quindi quanto più piccole sono le dimensioni dei problemi che l'algoritmo è in grado di risolvere in tempi ragionevoli (già per $p = 4$ la crescita è piuttosto rapida). Tuttavia la crescita polinomiale è sempre preferibile ad una esponenziale.

Domanda

Data una classe di problemi di ottimizzazione combinatoria, possiamo sempre trovare un algoritmo con complessità polinomiale che ne risolva le istanze?

Risposta: forse, ma è molto improbabile.

Vedremo di chiarire nel seguito il senso di questa risposta introducendo qualche elemento di *teoria della complessità*. Prima però prendiamo in esame algoritmi di risoluzione per alcuni dei problemi di ottimizzazione precedentemente discussi.

Algoritmi per SHORTEST PATH

Per questo problema abbiamo presentato due algoritmi:

- Algoritmo di Dijkstra: valido solo se $d_{ij} \geq 0 \; \forall \; (i,j) \in A$. Restituisce i cammini minimi tra un nodo fissato $s \in V$ e tuuti gli altri nodi del grafo.
- Algoritmo di Floyd-Warshall: valido anche per distanze negative *a patto che non ci siano cicli di lunghezza negativa*. Restituisce i cammini minimi tra tutte le coppie di nodi del grafo *se il grafo non contiene cicli a costo negativo*. In quest'ultimo caso restituisce un ciclo a costo negativo.

Complessità algoritmi

L'algoritmo di Dijkstra richiede un numero di operazioni $O(n^2)$.

L'algoritmo di Floyd-Warshall richiede un numero di operazioni $O(n^3)$.

Algoritmi visti per MST

- Algoritmo greedy (complessità $O(\mid E \mid \log(\mid E \mid))$);
- algoritmo MST-1 (complessità $O(\mid V \mid^2)$);
- algoritmo MST-2 (complessità $O(\mid E \mid \log(\mid V \mid))$).

La classe $\mathcal{P}$

Definizione *Dato un problema di ottimizzazione R, diciamo che questo appartiene alla classe $\mathcal{P}$ se e solo se esiste un algoritmo $\mathcal{A}$ di complessità polinomiale che lo risolve.*

Problemi in $\mathcal{P}$

Alla classe $\mathcal{P}$ appartengono, per esempio, i problemi SHORTEST PATH e MST per i quali abbiamo presentato algoritmi di complessità polinomiale che li risolvono.

La classe NP

Definizione *La classe NP contiene tutti i problemi di ottimizzazione per i quali, nota la soluzione ottima, il valore ottimo può essere calcolato in tempo polinomiale.*

Esempi

- `CLIQUE` data una clique di cardinalità massima, il valore ottimo si ottiene semplicemente calcolando la sua cardinalità.
- `TSP` Dato un circuito hamiltoniano ottimo $i_1 \to \cdots \to i_n \to i_{n+1} \equiv i_1$, il valore ottimo è la sua lunghezza $\sum_{j=1}^{n} d_{i_j i_{j+1}}$, che si calcola in un tempo $O(n)$ (somma delle n distanze).

Osservazione

Va notato che tutti i problemi in $\mathcal{P}$ fanno parte anche di $\mathcal{NP}$, ovvero

$$\mathcal{P} \subseteq \mathcal{NP}.$$

Per i problemi in $\mathcal{P}$ si ha infatti che non è neppure necessario che qualcuno ci dia una soluzione ottima per calcolare il valore ottimo in tempo polinomiale: è sufficiente utilizzare un algoritmo di complessità polinomiale per questi problemi (che esiste per definizione di classe $\mathcal{P}$).

Trasformazione in tempo polinomiale

Definizione *Dati due problemi di ottimizzazione R_1 e R_2, diciamo che R_1 è trasformabile in tempo polinomiale in R_2 se e solo se ogni istanza di R_1 di dimensione k può essere risolta risolvendo un'istanza di R_2 di dimensione al più $p(k)$ per un qualche polinomio p.*

Osservazione

Se R_1 è trasformabile in tempo polinomiale in R_2 e R_2 è risolvibile in tempo polinomiale, allora anche R_1 è risolvibile in tempo polinomiale.

Dimostrazione

Sia k la dimensione di un'istanza di problema R_1 e $p(k)$ la dimensione della istanza equivalente di R_2.

Ogni istanza di R_2 di dimensione h viene risolta in un tempo polinomiale $q(h)$.

Quindi, per risolvere l'istanza di R_2 attraverso cui si può risolvere l'istanza di R_1 considerata, abbiamo bisogno di un tempo $q(p(k))$, che è un polinomio rispetto a k.

I problemi NP-completi

Diciamo che un problema R è $\mathcal{NP}$-completo se:

- $R \in \mathcal{NP}$;
- per ogni problema $Q \in \mathcal{NP}$, esiste una riduzione polinomiale di Q in R.

Osservazione

Se esistesse un problema $\mathcal{NP}$-completo risolvibile in tempo polinomiale (o, equivalentemente, appartenente a $\mathcal{P}$), allora si dovrebbe avere che $\mathcal{P} = \mathcal{NP}$.

Abbiamo visto che $\mathcal{P} \subseteq \mathcal{NP}$ e possiamo chiederci ora se $\mathcal{P} = \mathcal{NP}$, oppure se esistono problemi in NP che non sono risolvibili in tempo polinomiale, cioè $\mathcal{P} \neq \mathcal{NP}$.

In realtà con le conoscenze attuali non si può ancora rispondere a tale domanda. Tuttavia tra le due possibili risposte quella che si ritiene la più probabile è che $\mathcal{P} \neq \mathcal{NP}$.

Continua

In base alla definizione di problemi $\mathcal{NP}$-completi al fatto che non sia chiaro se $\mathcal{P} = \mathcal{NP}$ oppure no, possiamo dire che non sono *al momento* noti algoritmi di complessità polinomiale in grado di risolvere problemi $\mathcal{NP}$-completi.

Inoltre, se, come si ritiene, $\mathcal{P} \neq \mathcal{NP}$, allora non esisterebbero algoritmi di complessità polinomiale per tali problemi.

Ne risulta quindi che la classe dei problemi $\mathcal{NP}$-completi è una classe di problemi "difficili" nel senso che, a meno che non sia $\mathcal{P} = \mathcal{NP}$, non possiamo risolverli in tempo polinomiale.

Esempi problemi $\mathcal{NP}$-completi

I problemi `CLIQUE`, `KNAPSACK`, `TSP` sono $\mathcal{NP}$-completi.

Problemi di approssimazione

Sia data un problema di ottimizzazione con istanze (f, S) tali che

$$\forall\, x \in S \;:\quad f(x) \geq 0.$$

Per ogni istanza sia x^* la soluzione ottima dell'istanza e sia

$$opt = f(x^*)$$

il valore ottimo dell'istanza.

Continua

Definizione *Per i problemi di massimo il problema di ε-approssimazione, $\varepsilon \geq 0$, consiste nel determinare un punto $\overline{x} \in S$ tale che*

$$\frac{opt}{f(\overline{x})} \leq 1 + \varepsilon.$$

Per i problemi di minimo il problema di ε-approssimazione consiste nel determinare un punto $\overline{x} \in S$ tale che

$$\frac{f(\overline{x})}{opt} \leq 1 + \varepsilon.$$

In entrambi i casi il punto $\overline{x}$ viene definito soluzione ε-approssimata dell'istanza.

Continua

Per $\varepsilon = 0$ il problema coincide con quello di ottimizzazione.

Ma per $\varepsilon > 0$ si richiede qualcosa di meno rispetto al problema di ottimizzazione: non si cerca la soluzione ottima ma una soluzione che non si discosti troppo da quella ottima.

In particolare, si ricava che in una soluzione ε-approssimata il valore f in corrispondenza di tale soluzione differisce, sia per i problemi di massimo che per quelli di minimo, per al più εopt dal valore ottimo opt dell'istanza.

Chiaramente, tanto maggiore è il valore di ε, quanto minore è la precisione garantita da una soluzione ε-approssimata.

Algoritmi di approssimazione

Un algoritmo $\mathcal{A}_\varepsilon$ si definisce *algoritmo di ε-approssimazione* per un problema di ottimizzazione, se risolve il problema di ε-approssimazione associato ad ogni istanza del problema di ottimizzazione.

Ma allora ...

... dato un problema $\mathcal{NP}$-completo, qual è la complessità dei corrispondenti problemi di ε-approssimazione per diversi possibili valori di ε?

Possiamo riconoscere quattro diversi possibili casi in ordine crescente di difficoltà.

I quattro gradi di difficoltà

- **Caso 1** Per ogni $\varepsilon > 0$ esiste un algoritmo di ε-approssimazione che richiede tempi polinomiali sia rispetto alla dimensione delle istanze, sia rispetto all'inverso $\frac{1}{\varepsilon}$ della precisione richiesta. In tal caso si dice che il problema ammette uno *schema di approssimazione completamente polinomiale* (FPTAS).
- **Caso 2** Per ogni $\varepsilon > 0$ esiste un algoritmo di ε-approssimazione che richiede tempo polinomiale rispetto alla dimensione delle istanze ma esponenziale rispetto all'inverso $\frac{1}{\varepsilon}$ della precisione richiesta. In tal caso si dice che il problema ammette uno *schema di approssimazione polinomiale* (PTAS).

- **Caso 3** Per valori piccoli di ε, anche il problema di ε-approssimazione è $\mathcal{NP}$-completo, mentre per valori di ε più elevati è risolvibile in tempo polinomiale.
- **Caso 4** Per ogni valore di ε il problema di ε-approssimazione è $\mathcal{NP}$-completo.

Tra i problemi visti

KNAPSACK rientra nel Caso 1: esiste un algoritmo di approssimazione per tale problema, denominato algoritmo *scaling-rounding*, di complessità $O\left(\frac{n}{\varepsilon^2}\right)$.

TSP rientra nel Caso 4 (anche nel sottocaso simmetrico): per esso non è possibile risolvere in tempi polinomiali, a meno che non sia $\mathcal{P} = \mathcal{NP}$, neppure il problema di ε-approssimazione per *ogni* valore di ε.