Business Analytics

1 Introduzione alle decisioni in condizioni di incertezza

Si distinguono tre principali tipologie di *Business Analytics*:

- **Business Analytics descrittiva**: ha l'obiettivo di analizzare e sintetizzare i dati storici al fine di comprendere che cosa è accaduto in passato (e.g. analisi di fenomeni osservati come l'overbooking nel trasporto aereo).
- **Business Analytics predittiva**: mira a stimare eventi o variabili future sulla base di modelli statistici o probabilistici, utilizzando informazioni storiche e ipotesi sul comportamento del sistema (e.g. distinzione tra *forecasting* e *prediction*).
- **Business Analytics prescrittiva**: si concentra sul supporto alle decisioni, indicando quale azione dovrebbe essere intrapresa per ottimizzare una misura di prestazione in presenza di vincoli e incertezza (e.g. problemi di *operations management*).

1.1 Motivazione: perché considerare l'incertezza

Previsioni puntuali e decisioni Un punto di partenza naturale nello studio delle decisioni in condizioni di incertezza è la **costruzione di una previsione puntuale** di una variabile aleatoria. Sia X una v.a. reale, con distribuzione di probabilità nota. Una **previsione puntuale** consiste nella scelta di un valore $x \in \mathbb{R}$, che rappresenta una **stima ex ante** della realizzazione futura di X. La **qualità di una previsione** deve essere misurata in funzione del **costo associato all'errore** di previsione, che dipende dalla discrepanza tra il valore scelto x e la realizzazione effettiva di X.

Errore quadratico medio (MSE) Una scelta naturale per misurare il **costo dell'errore di previsione è la perdita quadratica**. In questo caso, il costo associato alla previsione puntuale x è $(X-x)^2$. Il *problema decisionale* consiste quindi nel **minimizzare l'errore quadratico medio**: $\mathbb{E}\big[(X-x)^2\big]$. Sviluppando il valore atteso, si ottiene

$$MSE(x) \doteq \mathbb{E}\big[(X-x)^2\big] = \mathbb{E}[X^2] - 2x\,\mathbb{E}[X] + x^2.$$

La condizione di ottimalità del primo ordine implica che la *previsione ottima coincide con il valore atteso della variabile aleatoria*, $x^* = \mathbb{E}[X]$. Questo risultato mostra che l'uso del valore atteso come previsione puntuale è ottimale solo sotto l'ipotesi di una perdita quadratica.

Perdita assoluta e penalità asimmetriche Un criterio alternativo consiste nel misurare l'errore di previsione tramite la **deviazione assoluta**, $\mathbb{E}\big[|X-x|\big]$. In questo caso, la soluzione ottima è data dalla **mediana della distribuzione** di X, anziché dal valore atteso.
Nelle applicazioni economiche, errori di previsione positivi e negativi possono avere impatti diversi. È quindi naturale introdurre una **funzione di perdita asimmetrica** del tipo $L(x,X) = c_u(X-x)^+ + c_o(x-X)^+$, dove $(\cdot)^+ = \max\{0,\cdot\}$, mentre c_u e c_o rappresentano rispettivamente il **costo di sottostima e di sovrastima**. Il problema decisionale associato è

$$\min_{x\in\mathbb{R}} \ \mathbb{E}\big[c_u(X-x)^+ + c_o(x-X)^+\big].$$

La soluzione ottima è un quantile della distribuzione di X, il cui livello dipende dal rapporto tra c_u e c_o.
Il **messaggio essenziale** è che una previsione puntuale, tipicamente basata sul valore atteso, non è sufficiente per prendere decisioni ottimali in condizioni di incertezza. La previsione ottimale dipende dalla funzione di perdita adottata e quindi dall'impatto economico dell'errore di previsione. In generale, non esiste una previsione "migliore" in senso assoluto, ma solo previsioni coerenti con uno specifico criterio decisionale.

Modelli decisionali in ambito business L'obiettivo non è la previsione di una variabile aleatoria, ma la **selezione di un vettore di decisioni** $x \in X$ che ottimizzi una funzione economica in presenza di fattori di rischio. Se ξ denota un vettore di variabili aleatorie che influenzano il risultato economico, il problema può essere formulato come un problema di ottimizzazione sotto incertezza. Se non si dispone di informazioni probabilistiche sui fattori di rischio e si conosce solo un insieme di incertezza U, il problema assume la forma di un **worst-case robust optimization problem**:

$$\boxed{\min_{x \in X} \max_{\xi \in U} f(x, \xi)}$$

Se invece i fattori di rischio sono modellati come variabili casuali con distribuzione nota, si ottiene un **problema di ottimizzazione stocastica**:

$$\boxed{\min_{x \in X} \mathbb{E}_{\mathbb{P}}\big[f(x, \tilde{\xi})\big]}$$

Questa formulazione mette in evidenza che, in generale, $\mathbb{E}\big[f(x, \tilde{\xi})\big] \neq f\big(x, \mathbb{E}[\tilde{\xi}]\big)$, e giustifica la necessità di modelli decisionali che tengano esplicitamente conto dell'incertezza.

1.2 Modelli decisionali statistici

Il modello classico del newsvendor Il modello classico del *newsvendor* rappresenta un problema di decisione sotto condizione di incertezza in cui una decisione deve essere presa prima dell'osservazione di una variabile aleatoria. Sia D una variabile casuale non negativa che rappresenta la **domanda**, con distribuzione di probabilità nota. Il decisore sceglie una **quantità** $q \geq 0$, a **prezzo** c prima di osservare la realizzazione di D. Ogni pezzo viene venduto a un **prezzo** $p > c$ durante la finestra di vendita, e ad un prezzo ridotto successivamente $p_u < c$.
L'intuizione potrebbe suggerire di porre semplicemente $q = \mathbb{E}\big[D\big]$. Ma questo risulta essere sbagliato. In realtà, il risultato non sorprende se introduciamo due tipi di costo:

- Se $q < D$, avremo un costo opportunità $m = p - c$, cioè il margine di profitto per la parte di domanda non soddisfatta (**shortfall**).
- Se $q > D$, avremo un costo dell'invenduto $c_u = c - p_u$, legata alla svendita della giacenza residue (**surplus**).

Consideriamo una **formulazione analitica** del problema, dove assumiamo che la domanda abbia distribuzione nel continuo con densità $f_D(x)$ nota. Il valore atteso del profitto è:

$$EP(q) \equiv \mathbb{E}\big[\pi(q, D)\big] = m\left(\int_0^q x f_D(x)\,dx + \int_q^{+\infty} q f_D(x)\,dx\right) - c_u \int_0^q (q - x) f_D(x)\,dx.$$

Teorema: regola di Leibniz Consideriamo una funzione di due variabili $g(q, x)$ e definiamo una funzione della sola q come

$$G(q) = \int_{h_1(q)}^{h_2(q)} g(q, x)\,dx.$$

Notiamo che anche gli estremi di integrazione sono funzioni di q. Sotto opportune ipotesi di continuità, la regola di Leibniz permette di scrivere:

$$\frac{dG}{dq}(q) = \int_{h_1(q)}^{h_2(q)} \frac{\partial g}{\partial q}(q, x)\,dx + g\big(q, h_2(q)\big)\, h_2'(q) - g\big(q, h_1(q)\big)\, h_1'(q).$$

Nel caso del profitto atteso del newsvendor, tramite la regola di Leibniz otteniamo

$$\frac{d\,\mathbb{E}[\pi(q, D)]}{dq} = m\left(q f_D(q) + \int_q^{+\infty} f_D(x)\,dx - q f_D(q)\right) - c_u \int_0^q f_D(x)\,dx$$

$$= m \int_q^{+\infty} f_D(x)\,dx - c_u \int_0^q f_D(x)\,dx = m\big(1 - F_D(q)\big) - c_u F_D(q) = 0,$$

dove $F_D(x) \doteq \mathbb{P}\{D \leq x\}$ è la funzione di distribuzione cumulativa della domanda. Ponendo a zero la derivata prima, ricaviamo immediatamente

$$F_D(q^*) = \frac{m}{m + c_u}.$$

Come verifica, risulta utile controllare la **derivata seconda**:

$$\frac{d^2\mathbb{E}[\pi(q, D)]}{dq^2} = -m f_D(q) - c_u f_D(q) < 0, \qquad \forall q,$$

in quanto la funzione di densità non può assumere valori negativi. Ciò dimostra la **concavità del profitto atteso rispetto a** q.

La **condizione di ottimalità** prescrive un valore di q^* tale da ottenere una data probabilità di non andare in *stockout*, che è una misura del **livello di servizio**. Per valori di m molto grandi rispetto a c_u, il livello di servizio ottimale tende al 100%, e quindi si acquista/produce molta merce, mentre esso si riduce se il margine è basso.

Esempio: newsvendor nel caso normale Se assumiamo domanda normale, con valore atteso μ e deviazione standard σ, le note proprietà dei quantili di una distribuzione normale ci permettono di scrivere $q^* = \mu + z_\beta \sigma$, dove z_β è il quantile della normale standard corrispondente al livello di probabilità

$$\beta = \frac{m}{m + c_u}.$$

È facile vedere che il solo caso in cui è ottimale ordinare il valore atteso μ della domanda si verifica per penalità simmetriche, $m = c_u$. Si potrebbe pensare che all'aumentare dell'incertezza della domanda si debba essere conservativi e ordinare meno del valore atteso. In realtà non è così in generale.

Esiste anche un altro modo, del tutto equivalente, di ricavare la regola di decisione, introducendo una **funzione di perdita**

$$L(q, D) = m(D - q)^+ + c_u(q - D)^+,$$

legata alla discrepanza tra q e D, dove $(x)^+ \doteq \max\{x, 0\}$.

Si può dimostrare che il profitto atteso e il valore atteso della perdita differiscono per una costante:

$$\mathbb{E}[\pi(q, D)] = m\mathbb{E}[D] - \mathbb{E}[L(q, D)].$$

Massimizzare il profitto atteso è quindi equivalente a minimizzare il valore atteso della funzione di perdita, che rappresenta una penalità legata all'impatto economico della discrepanza tra q e D. Se si scegliesse q come previsione della domanda, si dovrebbe scegliere una funzione di perdita opportuna:

- nel caso di una perdita quadratica, $L(q, D) = (q - D)^2$, è facile dimostrare che la soluzione ottima sarebbe $q^* = \mathbb{E}[D]$;
- nel caso di una perdita data dalla deviazione assoluta, $L(q, D) = |q - D|$, usando la regola di Leibniz è facile dimostrare che la soluzione ottima è data dalla **mediana della domanda**, ovvero il quantile al livello di probabilità 50%.

Nel caso del newsvendor, la penalità è lineare a tratti, come la funzione valore assoluto, ma le due pendenze non sono simmetriche.

Espressione generale del profitto e riscrittura con costi di underage/overage La formulazione tramite funzione di perdita può essere affiancata da una scrittura esplicita del profitto, utile per evidenziare la decomposizione in termini di *underage* e *overage*. L'espressione generale del profitto è

$$\pi(q, d) = -cq + p \min(q, d) + r \max(q - d, 0),$$

dove $(x)^+ \doteq \max\{x, 0\}$. Usando l'identità

$$q = \min(q, d) + (q - d)^+,$$

il profitto può essere riscritto come

$$\pi(q, d) = c_u \min(q, d) - c_o(q - d)^+,$$

dove definiamo il costo di underage $c_u = p - c$ e il costo di overage $c_o = c - r$.

Se l'incertezza sulla domanda è modellata da una variabile casuale continua con densità $f_D(x)$, il profitto atteso è

$$\mathbb{E}[\pi(q, D)] = c_u \left(\int_0^q x f_D(x)\, dx + \int_q^{+\infty} q f_D(x)\, dx \right) - c_o \int_0^q (q - x) f_D(x)\, dx.$$

Applicando nuovamente la regola di Leibniz, la condizione di ottimalità del primo ordine porta a

$$c_u\big(1 - F_D(q)\big) - c_o F_D(q) = 0,$$

da cui segue che la quantità ottima q^* soddisfa

$$F_D(q^*) = \frac{c_u}{c_u + c_o}.$$

Questa condizione è facile da interpretare: la quantità ottima da ordinare è un **quantile della distribuzione della domanda**, che dipende dall'economia del problema. Quando c_u è grande rispetto a c_o, la quantità ottima è elevata; quando i due coefficienti di penalità sono uguali, la soluzione ottima coincide con la mediana della distribuzione della domanda.

Modelli con vincoli probabilistici (chance-constrained models) L'idea alla base dei *chance-constrained models* è che un vincolo stocastico non debba necessariamente essere soddisfatto in ogni realizzazione dell'incertezza, ma solo con una probabilità sufficientemente elevata. In particolare, un vincolo del tipo

$$g(x, \tilde{\xi}) \leq 0$$

è considerato accettabile se risulta soddisfatto con probabilità almeno pari a un livello prefissato. Sono considerati vincoli probabilistici individuali

$$\mathbb{P}\{g_j(x, \tilde{\xi}) \leq 0\} \geq 1 - \alpha_j, \qquad j \in [m],$$

e un vincolo probabilistico congiunto

$$\mathbb{P}\{g(x, \tilde{\xi}) \leq 0_m\} \geq 1 - \alpha,$$

dove g è una funzione vettoriale. L'intuizione potrebbe suggerire che, se le funzioni g_j sono convesse rispetto a x per ogni realizzazione di $\tilde{\xi}$, allora anche il vincolo probabilistico dovrebbe preservare la convessità. Tuttavia, come mostrato nelle slide tramite un esempio di vincoli di capacità, questo non è vero in generale: l'insieme ammissibile di un problema con vincoli probabilistici può risultare non convesso, in quanto definito come unione di insiemi ammissibili associati a diversi scenari. Per questo motivo, i modelli chance-constrained possono risultare difficili da trattare dal punto di vista computazionale e spesso vengono approssimati tramite formulazioni di tipo robusto. Nonostante ciò, essi rappresentano uno strumento naturale per modellare problemi statici in cui non è possibile adattare le decisioni dopo la realizzazione dell'incertezza.

1.3 Proprietà di convessità dei modelli di programmazione stocastica

2 Elementi di complessità computazionale

Recap La **complessità computazionale** dipende dalla dimensione dell'istanza e può crescere in modo fattoriale, esponenziale o polinomiale. Piccole variazioni nella struttura di un problema possono modificare radicalmente la sua complessità, come nel caso dei problemi di scheduling $1//L_{\max}$ e $1/r_i/L_{\max}$.

La **teoria della NP-completezza** introduce le classi P, NP, NPH e NPC e il concetto di riduzione polinomiale per confrontare la difficoltà dei problemi. I problemi NP-completi sono equivalenti in termini di complessità e l'esistenza di un algoritmo polinomiale per uno di essi implicherebbe $P = NP$.

La riduzione da subset-sum mostra che la versione decisionale di $1/r_i/L_{\max}$ è NP-completa e che il corrispondente problema di ottimizzazione è NP-difficile. **La codifica dei dati è cruciale**: l'algoritmo per il knapsack ha complessità pseudo-polinomiale, poiché dipende dal valore numerico dei dati e non dalla loro lunghezza binaria.

Complessità di problemi e algoritmi Si vuole caratterizzare la complessità in funzione della dimensione di un problema. Occorre distinguere la complessità degli algoritmi da quella dei problemi.
Nel caso di un algoritmo caratterizzato da un numero finito di passi, possiamo valutare (eventualmente nel caso peggiore) il **numero di operazioni elementari** in funzione della dimensione n del problema.

- **Algoritmi enumerativi**: algoritmi che considerano tutte le **permutazioni** di n oggetti. La loro complessità è $O(n!)$ ed è certamente non praticabile per valori di n anche moderatamente grandi.
- **Algoritmi di assegnamento esaustivo**: algoritmi che valutano tutti gli **assegnamenti possibili** di n variabili binarie. In questo caso la complessità cresce in modo esponenziale ed è pari a $O(2^n)$.
- **Algoritmi polinomiali**: un tipico esempio è rappresentato dagli **algoritmi di ordinamento** di n oggetti. Gli algoritmi più semplici hanno complessità $O(n^2)$, mentre algoritmi più efficienti raggiungono complessità $O(n \log_2 n)$.

Nel caso di **algoritmi iterativi** che generano una sequenza di soluzioni, si può cercare di caratterizzare la velocità di convergenza (es., lineare o quadratica).

Complessità intrinseca di un problema Una questione più sottile si pone quando si vuole caratterizzare la complessità intrinseca di un problema.
Per comprendere la natura della questione, consideriamo un semplice **problema di scheduling di n job su macchina singola**. Indichiamo con p_j il tempo necessario per il job $j = 1, \ldots, n$, e con d_j, $j = 1, \ldots, n$ la sua data di consegna (**due date**).
La soluzione è una sequenza di job, ovvero una permutazione σ in cui $\sigma(k)$ è l'indice del job in posizione k. I tempi di completamento sono

$$\begin{aligned} C_{\sigma(1)} &= p_{\sigma(1)}, \\ C_{\sigma(k)} &= C_{\sigma(k-1)} + p_{\sigma(k)}, \quad k = 2, \ldots, n. \end{aligned}$$

Si vuole minimizzare la massima lateness,

$$L_{\max} \doteq \max_{j \in [n]} L_j,$$

dove $L_j \doteq C_j - d_j$. Tale problema viene indicato con la stringa $1//L_{\max}$.

Teorema (regola EDD – Earliest Due Date). Per il problema $1//L_{\max}$ esiste una soluzione ottima in cui $d_{\sigma(k)} \leq d_{\sigma(k+1)}$.
Dimostrazione. Supponiamo che esista una soluzione ottima in cui, per due job consecutivi in sequenza, prima $i = \sigma(k)$ e poi $j = \sigma(k+1)$, si abbia $d_i > d_j$. Indichiamo con C_i e $C_j = C_i + p_j$ i due tempi

di completamento nella soluzione corrente, per la quale abbiamo due valori di lateness $L_i = C_i - d_i$ e $L_j = C_j - d_j$.
Se scambiamo i due job, avremo $C'_j < C_j$ e $C'_i = C_j$. Per il job j, che anticipiamo, abbiamo $C'_j < C_j$ e $L'_j < L_j$, e quindi la lateness del job j non può che migliorare nella nuova soluzione. Per il job i, che viene spostato in avanti nella sequenza, abbiamo

$$L'_i = C'_i - d_i = C_j - d_i < C_j - d_j = L_j,$$

quindi il job i peggiora la sua lateness, che però non potrà essere peggio della vecchia lateness del job j. La nuova soluzione migliora quella precedente, contraddicendo l'assunzione di ottimalità.

Abbiamo quindi un algoritmo di complessità polinomiale per il problema. **Ma cosa accade se complichiamo leggermente il problema, introducendo dei tempi di rilascio (release time o ready time)** r_i**,** $i \in [n]$**, dei job?** Per il problema $1/r_i/L_{\max}$ non sono noti algoritmi di complessità polinomiale, ed è facile costruire controesempi alla regola EDD (da adattare comunque alla disponibilità di job).
Esistono algoritmi **branch-and-bound** per il problema $1/r_i/L_{\max}$ (quindi complessità esponenziale). Non si conosce un algoritmo di complessità polinomiale per questo problema di ottimizzazione combinatoria (e per molti altri), ma neppure è stato dimostrato che esso non possa esistere.

2.1 Caratterizzazione della complessità di problemi: classi P, NPC e NPH

Esiste una classe molto ampia di problemi di ottimizzazione per cui non sono disponibili algoritmi di complessità polinomiale. Tuttavia, la questione dell'esistenza o meno di un algoritmo di complessità polinomiale per essi rimane aperta.
La **teoria della NP-completezza** ci permette di dare una risposta parziale alla questione, mostrando come questi problemi siano equivalenti tra di loro, nel senso che un algoritmo di complessità polinomiale per anche uno solo di essi fornirebbe un algoritmo di complessità polinomiale per tutti i problemi di una classe molto ampia.
Il fatto che decenni di ricerca sulla soluzione di tutti questi problemi non abbiano prodotto un algoritmo polinomiale **suggerisce che esso in effetti non esiste**.
Occorre distinguere **problemi di decisione** e **problemi di ottimizzazione**. Esempi di problema di decisione sono i seguenti.

Problema K_0 **(subset sum)** Dati $n+1$ numeri interi positivi $a_1, a_2, \ldots, a_n$ e b, esiste un sottoinsieme $J \subseteq [n]$ tale che $\sum_{i \in J} a_i = b$?

Problema LS_0 Dato un **problema di lot sizing multiprodotto**, con tempi e costi di setup, esiste una soluzione ammissibile rispetto al soddisfacimento della domanda e ai vincoli di capacità produttiva?
Data una specifica istanza di un problema di decisione, la risposta è sì oppure no.
Dal punto di vista teorico, è più agevole trattare problemi di decisione, **ma è facile vedere il legame tra problemi di ottimizzazione e problemi di decisione**. Dato un problema di ottimizzazione $\min_{x \in S} f(x)$, **possiamo definirne una versione decisionale**, scegliendo un numero k e chiedendoci se esiste $x \in S$ tale che $f(x) \leq k$, dove k è un numero intero. Indichiamo con PO il problema di ottimizzazione, e con PD il corrispondente problema di decisione.
Se abbiamo a disposizione un algoritmo efficiente per PO, allora possiamo risolvere in modo efficiente anche PD: basta risolvere il problema di ottimizzazione e verificare se $f(x^*) \leq k$. Questo ci permette di scrivere $PD \to PO$, nel senso che **il problema di decisione può essere ricondotto al problema di ottimizzazione**.
Non è detto che tale trasformazione sia conveniente, ma possiamo escludere che PO sia facile se PD è difficile, perché un ipotetico algoritmo efficiente per PO risolverebbe anche facilmente PD.
Quindi, per dimostrare che un problema di ottimizzazione è difficile, può bastare dimostrare che è difficile il corrispondente problema di decisione.
D'altro canto, un algoritmo di decisione efficiente potrebbe, in certi casi, essere utilizzato per risolvere il

problema di ottimizzazione. Se la funzione di costo in PO assume valori interi non negativi, e abbiamo un upper bound U sul costo ottimo, possiamo applicare una procedura di bisezione.
Indichiamo con P la **classe dei problemi di decisione per cui esiste un algoritmo di complessità polinomiale**, in grado di risolvere tutte le possibili istanze del problema. Con questo vogliamo dire che il numero di passi, e quindi la complessità temporale dell'algoritmo è limitata superiormente da una funzione polinomiale dello spazio di memoria necessario per descrivere ogni istanza del problema. Un tale algoritmo è in grado di fare due cose: **generare una soluzione** e **verificarne la correttezza**. Esistono problemi, come K_0, per cui la prima parte del compito è difficile, ma la seconda no. Se enumeriamo, mediante un albero di ricerca, tutti i possibili sottoinsiemi J, possiamo verificare se una specifica istanza ha risposta positiva o negativa, ma chiaramente tale algoritmo ha **complessità esponenziale**.
Tuttavia, se disponessimo di un ipotetico **calcolatore non deterministico**, in grado di eseguire un numero infinito di processi di calcolo in parallelo, saremmo in grado di risolvere il problema in tempo polinomiale.

Classe NP Si definisce **classe** NP l'insieme dei problemi di decisione le cui istanze che hanno risposta positiva sono verificabili in tempo polinomiale.
Per definizione, $P \subseteq NP$. Una questione meno ovvia è se valga $P \equiv NP$ o $P \subset NP$ in senso stretto. È ragionevole, da questo punto di vista, cercare di caratterizzare la sottoclasse dei problemi più difficili in NP.

Riduzione in tempo polinomiale Siano P e Q due problemi di decisione, per cui ogni istanza I_P di P può essere trasformata in tempo polinomiale in un'istanza I_Q di Q tale che I_P ha risposta positiva se e solo se I_Q ha risposta positiva. Diremo che P è riducibile in tempo polinomiale a Q, e useremo la notazione $P \prec Q$.) La notazione $P \prec Q$ sottolinea che la complessità di P non è maggiore della complessità di trasformare P in Q e poi risolvere Q,

$$\text{compl}(P) \leq \text{compl}(Q) + \text{compl}(P \to Q).$$

Se la trasformazione ha una complessità trascurabile, la riduzione di P a Q mostra che Q non è più facile di P. Se P è difficile e $P \prec Q$, Q non può essere facile. Altrimenti, potremmo trasformare un'istanza di P in una di Q, e poi applicare l'algoritmo efficiente per Q.

Problemi NP-difficili Un problema di decisione P è detto **NP-difficile** (*NP-hard*) se ogni problema nella classe NP è riducibile a P. Indichiamo con NPH la classe dei problemi NP-difficili.

Problemi NP-completi Un problema di decisione P è detto NP-completo se è in Np ed è NP-hard. Indichiamo con NNPC la classe dei problemi NP-completi. Le implicazioni pratiche di tali definizioni sono:

1. Un problema NP-difficile non è più facile di un problema qualsiasi in NP.
2. La classe NPC è la classe dei problemi più difficili in NP.

Per dimostrare che un problema di decisione P è NP-completo, occorre dimostrare:

- Che P è in NP;
- Che un problema NP-completo Q può essere ridotto in tempo polinomiale a P.

Osserviamo che, dato che Q è NP-completo, $P \prec Q$, e se trascuriamo la complessità della trasformazione, questo implica $\text{compl}(P) \leq \text{compl}(Q)$. Ma il secondo passo della dimostrazione di NP-completezza, ovvero dimostrare che $Q \prec P$, implica anche che $\text{compl}(Q) \leq \text{compl}(P)$. Le due disuguaglianze, sempre a meno della complessità della trasformazione, mostrano che $\text{compl}(Q) = \text{compl}(P)$.

In altre parole, **i problemi della classe NPC sono equivalenti in termini di complessità computazionale, e un algoritmo polinomiale per uno di essi permetterebbe di risolverli tutti in**

tempo polinomiale. Avremmo quindi $P = NP$, ipotesi non troppo plausibile a causa dell'equivalenza di una vasta classe di problemi per i quali non è noto un algoritmo di complessità polinomiale, nonostante essi siano stati oggetto di ampio studio nel corso degli anni.
Se accettiamo l'ipotesi $P \neq NP$, possiamo rappresentare le relazioni tra le classi P, NP e NPC come in figura. Essa ipotizza una gerarchia per cui la classe P sarebbe la classe dei problemi più facili in NP, e NPC quella dei problemi più difficili.
Tutti i problemi in NP si possono trasformare nel problema $Q \in NPC$. Se $P \in NP$, per dimostrare che $P \in NPC$, occorre trasformare Q in P.

Se dimostriamo che un problema P è in NPC, questo può a sua volta essere trasformato in altri problemi, permettendoci di ampliare la classe dei problemi noti in NPC. Il punto critico, chiaramente, è trovare l'innesco della catena, ovvero il primo problema in NPC, al quale tutti i problemi in NP possono essere ricondotti.

Il **teorema di Cook** dimostra che il problema della soddisfacibilità soddisfa i requisiti necessari e ci fornisce la soluzione.

Teorema di Cook. Data una formula Booleana in forma canonica disgiuntiva, decidere se esiste un assegnamento di valori ai suoi elementi che la rende vera. Per esempio, la formula (A or B) and (not(A) or C); definita rispetto alle variabili Booleane A, B e C, è soddisfatta se B e C sono entrambe vere. Al contrario, (A or B) and (not(A) or B) and (not(B)) non può essere soddisfatta da alcun assegnamento di verità alle variabili.

A partire dal problema della soddisfacibilità, si può ricavare per riduzioni polinomiali successive una famiglia crescente di problemi NP-completi, compreso il problema K_0, che può essere considerato come un cugino in versione decisionale del problema knapsack.
Il fatto che tale problema faccia parte della classe NPC ci permette di dimostrare il teorema seguente, che risolve la questione da cui siamo partiti.

Teorema. Consideriamo una versione decisionale del problema $1/r_i/L_{\max}$: dati i tempi di rilascio r_i, le date di consegna d_i e i tempi di lavorazione p_i, tutti a valori interi positivi, per n job J_i, $i \in [n]$, esiste una soluzione in cui nessun job è completato in ritardo? Tale problema di decisione è NP-completo.
Dimostrazione. Il problema è chiaramente in NP, poiché per una data sequenza è facile verificare se i job vengono completati in tempo rispetto alle due date.
Dati gli interi positivi a_j, $j \in [n]$, creiamo n job J_j con parametri

$$r_j = 0, \quad p_j = a_j, \quad d_j = 1 + \sum_{k \in [n]} a_k, \qquad j \in [n].$$

Creiamo un ulteriore job J_0 con parametri $r_0 = b, \quad p_0 = 1, \quad d_0 = b + 1$. Perché tutti i job rispettino la data di consegna, è necessario che il job J_0 inizi la lavorazione al tempo $t = b$. Inoltre, dato che la data di consegna degli altri job è pari alla somma di tutti i tempi di lavorazione, la soluzione non può presentare periodi di tempo in cui la macchina è ferma, prima di avere completato l'intero insieme di job.
Questo richiede che sia possibile individuare un sottoinsieme J di job da schedulare prima di J_0, in modo tale che

$$\sum_{j \in J} p_j = r_0.$$

Tale insieme risolve il problema *subset-sum*.

2.2 Dai problemi di decisione ai problemi di ottimizzazione

Il teorema dimostra che un problema di decisione legato al problema di ottimizzazione $1/r_i/L_{\max}$ è NP-completo, **ma cosa possiamo dire del problema di ottimizzazione stesso?**
Per definizione, la classe NPC contiene solo problemi di decisione. Possiamo però estendere le classi P e NPH, includendo in esse anche problemi di ottimizzazione.

I **problemi di ottimizzazione** per cui è noto un algoritmo di complessità polinomiale stanno in P. Nella classe NPH possiamo includere problemi di ottimizzazione ai quali possiamo ricondurre un corrispondente problema di decisione. **La versione decisionale di $1/r_i/L_{\max}$ si riduce chiaramente al problema di ottimizzazione.**
Inoltre, nella dimostrazione abbiamo assunto che i dati fossero numeri interi. Ma il problema a numeri interi può evidentemente essere ridotto al problema generale. Possiamo quindi affermare che il problema di scheduling $1/r_i/L_{\max}$ è NP-difficile.

L'impatto della codifica di un problema Nella trattazione ci siamo limitati alle classi fondamentali e siamo stati piuttosto informali e imprecisi. Non possiamo però fare a meno di considerare l'impatto del modo in cui si codifica un problema. Sarebbe infatti errato, per esempio, associare a un problema knapsack una dimensione pari al numero di oggetti. La dimensione si riferisce a una codifica binaria che comprende tutti i dati del problema. Per mostrare la rilevanza di ciò, **consideriamo un classico algoritmo di programmazione dinamica** per la soluzione del problema knapsack:

$$\begin{aligned} \max \quad & \sum_{k=1}^{n} v_k x_k \\ \text{s.t.} \quad & \sum_{k=1}^{n} w_k x_k \leq B \\ & x_k \in \{0,1\}, \quad k = 1, \ldots, n. \end{aligned}$$

Definiamo la funzione valore

$$V_k(s) := \text{valore del sottoinsieme ottimale tra gli oggetti } \{k, k+1, \ldots, n\},$$

quando la capacità residua è s. In sostanza, la funzione valore assume che siano già state fatte scelte di inserimento o meno degli oggetti da 1 a $k-1$; a valle di tale selezione, abbiamo una capacità residua s, e ci chiediamo come utilizzarla al meglio per le scelte rimanenti. Se i dati w_k e B del problema sono interi, lo sarà anche la capacità residua s.
Per risolvere il problema, ovvero trovare il valore di $V_1(B)$, si applica una relazione ricorsiva:

$$V_k(s) = \begin{cases} V_{k+1}(s), & 0 \leq s < w_k, \\ \max\{V_{k+1}(s),\, V_{k+1}(s - w_k) + v_k\}, & w_k \leq s \leq B. \end{cases}$$

Si tratta di una equazione funzionale con condizione terminale:

$$V_n(s) = \begin{cases} 0, & 0 \leq s < w_n, \\ v_n, & w_n \leq s \leq B. \end{cases}$$

Occorre tabulare tutte le funzioni $V_k(s)$, $k = 1, \ldots, n$, per valori interi di s, che assume valori nel range da 0 a B. Pertanto, tale algoritmo ha complessità $O(nB)$.
Questo non implica che il problema knapsack abbia complessità polinomiale: per rappresentare il valore B in aritmetica binaria bastano $\log_2 B$ bit. Quindi **l'algoritmo, rispetto a tale codifica binaria, ha complessità esponenziale**. Se si utilizzasse un **computer con una codifica unaria**, l'algoritmo che abbiamo considerato avrebbe **complessità polinomiale**. Si dice infatti che un algoritmo di questo tipo è *pseudo-polinomiale*.