CAP 2
ELEMENTI DI COMPLESSITÀ COMPUTAZIONALE

Classificazione dei *prob di ottimizzazione* in base alla **difficoltà intrinseca del prob**, *indipendentemente* dall'**algo**.

Obiettivi principali

- distinguere **complessità** del *prob* vs *algo*
- introdurre **prob di decisione vs ottimizzazione**
- definire le classi **P**, **NP**, **NPH**, **NPC**

COMPLESSITÀ INTRINSECA: SCHEDULING

Problema di scheduling (min,max) su macchina singola. Sequenziamento di job su una macchina per controllare il max ritardo rispetto alle due date.

$$C_{\sigma(1)} = p_{\sigma(1)},$$
$$C_{\sigma(k)} = C_{\sigma(k-1)} + p_{\sigma(k)}, \quad k = 2, \dots, n.$$

$$L_{\max} \doteq \max_{j\in[n]} L_j \qquad L_j \doteq C_j - d_j.$$

Teorema (regola EDD – Earliest Due Date). Per il prob *1\\Lmax* esiste una *sol ottima* in cui i job sono ordinati per due date crescenti. $d_{\sigma(k)} \leq d_{\sigma(k+1)}$

➡️ *algo polinomiale*

➡️ *prob computazionalmente trattabile*

Realise Time. L'introduzione dei tempi di rilascio (*1\ri\Lmax*) vincola l'avvio dei job e rende non più ottima la regola EDD, aumentando la compl intrinseca del prob. **Non esistono** algo di compl polinomiale che lo risolvono, solo **branch-and-bound**.

PROBLEMI DI DECISIONE VS OTTIMIZZAZIONE

- **prob di decisione (PD)**: risposta binaria *(sì / no)*
- **prob di ottimizzazione (PO)**: ricerca migliore sol rispetto a una f obiettivo $\min_{x\in S} f(x)$

Legame tra PO e PD. Dato un PO, si definisce un PD scegliendo un valore k e chiedendo se esiste x in S tc f(x)<k.

PD→PO. Se si ha a disposizione un algo efficiente per PO, è possibile risolvere in modo efficiente PD

➡️ *se PD è difficile ⇒ PO non può essere facile*

➡️ *per dimostrare che un prob di ottimizzazione è difficile, è sufficiente dimostrare che è difficile il corrispondente prob di decisione*

SCHEDULING CON RELEASE TIMES

La versione decisionale del prob di scheduling 1/rj/Lmax è **NP-completa**.

- *mostra che l'intrattabilità nasce con l'introduzione dei tempi di rilascio*
- *PO è NP-difficile*

TEORIA DELLA NP-COMPLETEZZA

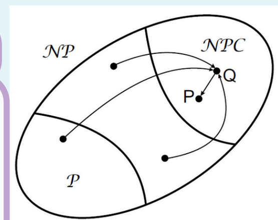
Esiste una vasta classe di *prob di ottimizzazione* per cui non sono noti algo polinomiali.

Idea centrale. La **teoria della NP-completezza** mostra che molti di questi prob sono equivalenti dal punto di vista *computazionale*.

➡️ *se uno solo ammettesse un algo pol tutti i prob della classe lo ammetterebbero*

Conseguenza fondamentale. Decenni di ricerca senza successo suggeriscono che

- *tali algo probabilmente non esistono*
- *la difficoltà è intrinseca, non dovuta a modelli*



CLASSI NPH – NPC

Classe NPH. Prob P tc ogni prob in NP è riduc a P.

➡️ *PO + PD*

Classe NPC. Prob P tc

- P è in NP **e** P è NPH

➡️ *problemi più difficili in NP, tutti equivalenti*

➡️ *per dimostrare che un PD P è NPC, occorre dimostrare che P è in NP e un prob NP-completo Q può essere ridotto in tempo polinomiale a P*

Teorema di Cook. Il prob della soddisfacibilità booleana è NP-completo.

$$(A \text{ or } B) \text{ and } (\text{not}(A) \text{ or } C)$$

CLASSI P – NP

Classe P. Prob di decisione per cui esiste un algo di compl polinomiale: *il numero di passi è limitato superiormente da una funzione polinomiale.* L'**algo**

- *trova una soluzione*
- *ne verifica la correttezza*

Classe NP. *Prob di decisione* le cui istanze che hanno risposta positiva sono verificabili in tempo polinomiale. Esiste un certificato polinomiale su un ***calcolatore non deterministico***.

RIDUZIONE POLINOMIALE

Un prob P è **riducibile** a Q se *ogni istanza di P può essere trasformata in tempo polinomiale in un'istanza di Q* con la stessa risposta.

- se P è difficile e **P<Q** ⇒ Q non può essere facile
- *la compl di P non è maggiore della compl di trasformare P in Q e poi risolvere Q*

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

IMPATTO CODIFICA (KNAPSACK)

Idea chiave. La compl dipende dalla codifica dell'input, non solo dal prob.

- *B* è **codificato in binario** ⇒ input di dimensione logB
- *O(nB)* è **esponenziale** nella dimensione dell'input

Pseudo-Polinomiale. L'algo, rispetto alla codifica binaria, ha *compl esponenziale*. Se si utilizzasse un **computer con una codifica unaria**, l'algo avrebbe *compl polinomiale*.

CAP 4
SISTEMI MRP – ERP – APPROCCIO JIT

Classificazione dei *sis di pianif* e *controllo della produz multilivello e con variabilità.*

Obiettivi principali

- distinguere **logiche push e pull**
- chiarire il ruolo di **variabilità**, **WIP** e **lead time**
- pianificazione **MRP** a *capacità infinita*
- approccio **Just-In-Time** (*Toyota*)

Idea chiave. Le *prestazioni* del sis produttivo dipendono dalla **variabilità** (propagata o controllata).

LIMITI ED EVOLUZIONE DEI SISTEMI MRP

Limiti degli approcci classici. I <u>modelli tradizionali</u> risultano inadeguati

- in *ambienti non make-to-stock (make-to-order, assemble-to-order)*
- presenza di *vincoli di capacità produttiva*
- *distinte base multilivello*

Effetto di amplificazione della variabilità. La **propagazione dei fabbisogni** lungo la distinta base può generare una *<u>forte amplificazione della variabilità</u>*, anche con domanda finale regolare.

Evoluzione dei modelli MRP. I <u>sis MRP</u> (***Material Requirements Planning***) nascono come ***risposta operativa al prob del lot-sizing multilivello***

➡️ assunzione di *capacità infinita*

➡️ utilizzo di *lead time fissati a priori*

Evoluzione verso MRPII ed ERP.

- **MRPII** (***Manufacturing***): introduce la *verifica dei <u>vincoli di capacità</u>*
- **RCCP** (***Rough Cut Capacity***): *verifica <u>aggregata</u>* e approssimata della *capacità*
- **CRP** (***Capacity Requirement***): *verifica <u>dettagliata</u>* capacità sulle <u>singole</u> risorse
- **ERP** (***Enterprise Resource***): integrazione *pianific con f <u>commerciali</u>* e *<u>finanziarie</u>*

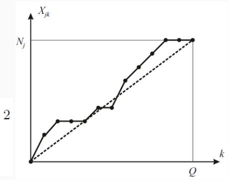
APPROCCIO JUST-IN-TIME

Definizione e obiettivo. Il *<u>Just-In-Time (JIT)</u>* mira a *ridurre la variabilità* alla fonte tramite *produzione livellata* (**production smoothing**), *lotti piccoli* e *frequenti* e ***<u>riduzione dei tempi di setup</u>***, con l'obiettivo di contenere WIP e lead time.

Logiche di controllo.

- **push**: *rilascio ordini da previsione*
- **pull**: produzione attivata *da domanda* reale
- **kanban**: *controllo pull locale a segnali*
- **CONWIP**: controllo *pull* con *WIP* globale

$$\sum_{k=1}^{Q}\sum_{j=1}^{M}\left(\frac{kN_j}{Q}-X_{jk}\right)^2$$



Goal chasing. Il *Toyota Goal Chasing* seleziona la ***sequenza produttiva che rende regolare il consumo dei componenti***, <u>minimizzando la distanza tra consumo ideale e consumo effettivo lungo il ciclo</u>.

Rotazione ciclica dei prodotti. I prodotti si *alternano* su una linea con periodo di rotazione.

$$p_iT_i = d_iT_c \qquad \Rightarrow \qquad T_i = \frac{d_i}{p_i}T_c$$

$$T_c \geq \sum_{i=1}^{N} s_i + \sum_{i=1}^{N} T_i$$

Il ***limite inferiore*** dipende dai *<u>tempi di setup</u>* e dal *<u>rapporto tra tassi di domanda e produzione</u>*, evidenziando un legame con i fenomeni di congestione.

$$T_c \geq \frac{\sum_{i=1}^{N} s_i}{1-\sum_{i=1}^{N}\frac{d_i}{p_i}}$$

LEGGE DI LITTLE

Prestazioni di shop floor. La **Factory Physics** descrive le prestazioni tramite *throughput*, *flow time* e *WIP*.

Legge di Little. Esprime il *legame strutturale* tra queste grandezze. $\mathrm{WIP} = \mathrm{throughput} \times \mathrm{flow\ time}$ $\quad L = \lambda(W_q + t_s)$

Modello a singola macchina e variabilità. In una singola macchina, l'*attesa in coda cresce con l'<u>utilizzazione</u>* e con la *<u>variabilità</u>* dei tempi di interarrivo e servizio. $u = \lambda/\mu$

$$W_q \approx \left(\frac{C_a^2 + C_s^2}{2}\right)\left(\frac{u}{1-u}\right)t_s$$

Buffering law. In presenza di variabilità, il sistema deve introdurre *buffer sotto forma di WIP, capacità* o *tempo o lead time.*

MPS e CRP

MPS. Il MPS è l'*input primario dell'MRP*, basato su *ordini cliente* e **forecasting**; può essere <u>validato</u> tramite **RCCP** e <u>strutturato a due livelli</u> in contesti ATO.

CRP. Il CRP *verifica a posteriori la capacità*; la correzione manuale è complessa e può generare lead time gonfiati e WIP, innescando un *circolo vizioso*.

NERVOSISMO

Nervosismo. *Piccole variazioni nel MPS producono grandi variazioni negli ordini pianificati <u>dovute a</u>*

- **lot-sizing** a *quantità variabile*
- **effetto di bordo**: instabilità da *rolling horizon*

Effetti. *Instabilità* del piano e *ordini urgenti.*

Mitigazione.

- **time fencing**: *congelamento* temporale MPS
- **firm planned orders**: ordini *non modificabili*

LOGICA MRP

Assunzione di capacità infinita. Il *vincolo di capacità <u>non è modellato</u>* ed è surrogato da *lead time fissati a priori*.

Lead time offsetting. Gli *ordini pianificati* sono <u>anticipati</u> nel tempo rispetto ai fabbisogni.

Record MRP. Per ogni <u>codice</u> e <u>periodo</u>

- *fabbisogni lordi*
- *magazzino disponibile* (*on-hand*)
- *ordini emessi* (*on-order*)
- *fabbisogni netti*
- *ordini pianificati*

➡️la <u>domanda</u> dei prodotti finiti è definita dal **MPS** (***Muster Production Schedule***)

➡️l'**MRP** procede <u>ricorsivamente lungo la distinta base</u>

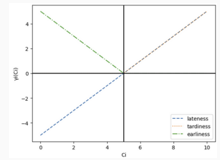
Ordini pianificati. <u>Non sono esecutivi</u>; al rilascio diventano ordini operativi e allocano giacenze.

Lot-sizing. Regola base *<u>lot-for-lot</u>*: produci esattamente ciò di cui hai bisogno.

MISURE DI PRESTAZIONE

Funzioni di penalità.

- **tempo di completamento** (*Ci*): istante di *fine dell'ultima operazione del job*
- **flow time** (*Fi*): Ci – ri, *tempo totale* trascorso nel sistema
- **lateness** (*Li*): Ci – di, *anticipo o ritardo rispetto alla due date*
- **tardiness** (*Ti*): max(Ci – di, 0), penalizza solo i *ritardi*
- **earliness** (*Ei*): max(di – Ci, 0), penalizza solo gli *anticipi*
- **indicatore di ritardo** (*Ui*): vale 1 se Ci > di, 0 altrimenti



Misure aggregate. Flow time totale, flow time totale pesato, massima lateness, tardiness totale pesata, makespan (*massimo dei Ci*), numero di job in ritardo.

Soluzioni equivalenti. Una sol ottima rispetto a una misura è ottima anche per un'altra; ***es*** lateness totale e flow time totale differiscono solo per una costante.

Misure regolari. F non decrescenti dei tempi di completamento Ci.

Misure non regolari. F non monotone in Ci, con penalità di earliness e tardiness.

- **schedul semiattiva**: ogni op è eseguita il più presto possibile
- **schedul attiva**: non esiste op anticipabile senza ritardarne un'altra

Notazione di Graham (alpha | beta | gamma). Layout delle *macchine*, *vincoli* aggiuntivi, misura di *prestazione*.

ALGORITMI DI SOL NELLO SCHEDULING

Algoritmi polinomiali (casi speciali).

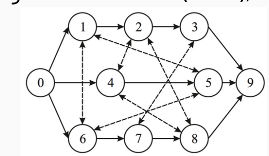
- **EDD**: ordinamento per *due date crescenti*; risolve 1\\Lmax
- **WSPT**: ordinamento per *wi/pi* decrescente; risolve 1\\wiCi
- **Johnson**: per F2\\Cmax; la sol ottima usa la *stessa sequenza sulle 2 macchine*

Regola ATC (Apparent Tardiness Cost). Assegna priorità combinando **peso del job**, durata dell'operazione e urgenza rispetto alla due date.

- se il ***job è in tempo*** la *priorità cresce* esponenzialmente
- se è in **ritardo** si riduce alla regola WSPT $\frac{w_i}{p_{ij}} \exp\left(-\frac{\left[d_i - t - p_{ij} - \sum_{q=j+1}^{m_i}(W_{iq}+p_{iq})\right]^+}{k\bar{p}}\right)$

Lookahead + ricerca locale.

- *beam search*: riduce la **miopia** delle regole di priorità
- *criticità*: evitare *minimi locali* (tabu, genetici), esplorare *grandi vicinati* (LNS), *evitare cicli*



Grafi disgiuntivi.

- **nodi**: operazioni + dummy iniziale/finale
- **archi congiuntivi**: precedenze tecnologiche del job
- **archi disgiuntivi**: capacità macchina (clique per macchina), da orientare
- **cammino critico**: lunghezza massima start→end = *makespan*

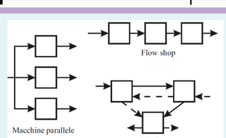
MACHINE SCHEDULING

Problemi di scheduling. Assegnazione di risorse a *job* nel *tempo*, rispettando ***vincoli*** *tecnologici*, di *capacità*, *precedenze*, ***tempi*** *di processo*, *due date*.

Soluzioni e diagrammi di Gantt. Una *sol* è definita dalle *sequenze di lavorazione sulle macchine* ed è visualizzata tramite *diagrammi di Gantt*, che *rappresentano graficamente l'allocazione temporale dei job*.

Tipi di flusso.

- **macchina singola**: una *sola* risorsa
- **macchine parallele**: identiche, correlate o scorrelate
- **flow shop**: *stesso ordine* di macchine
- **job shop**: cicli di lavorazione *diversi*
- **open shop**: *nessun ordine* prefissato



CAP 5
SCHEDULAZIONE in PRODUZ-SERVIZI

Schedulazione di job su risorse nel tempo, con *vincoli tecnologici e di capacità*.

Obiettivi principali

- **misure di prestazione**: f sui tempi di completamento, aggregate min-sum o min-max
- **classificazione dei prob**: *notazione di Graham*
- **compl computaz**: distinzione tra casi polinomiali (EDD, WSPT, Johnson) e prob NPH
- **strategie di soluzione**: uso *euristiche* e *shifting bottleneck* per decomporre *sis complessi*

PROCEDURA SHIFTING BOTTLENECH

Idea. Affrontare il problema *J//Cmax* **decomponendolo** in una sequenza di *sottoproblemi su singola macchina*, sfruttando il grafo disgiuntivo.

Approssimazione del makespan. Ottenuta tramite *teste e code delle operazioni lungo il cammino critico*, che stimano i tempi di rilascio e le scadenze locali.

Riduzione. Ogni macchina induce un problema 1/ri/Lmax, risolto in modo efficiente.

Identificazione del collo di bottiglia. La macchina con Lmax peggiore; la sua sequenza viene fissata e il processo iterato sulle restanti macchine.

MODELLO MILP J//Cmax

Nel modello MILP per J//Cmax solo *perturbazioni degli archi disgiuntivi sul cammino critico sono utili, poiché evitano la creazione di cicli.*

$$
\begin{aligned}
\min \quad & C_N & \\
\text{s.t.} \quad & C_j \geq C_i + p_j, & \forall (i,j) \in P,\\
& C_j \geq C_i + p_j - M(1 - x_{ij}), & \forall (i,j) \in D,\\
& C_i \geq C_j + p_i - M x_{ij}, & \forall (i,j) \in D,\\
& x_{ij} \in \{0,1\}, & \forall (i,j) \in D,\\
& C_i \geq p_i, & \forall i \in N.
\end{aligned}
$$