

Business Analytics – 2024/25

Schedulazione nella produzione e nei servizi

Prof. Paolo Brandimarte

Dip. di Scienze Matematiche – Politecnico di Torino

e-mail: paolo.brandimarte@polito.it

URL: staff.polito.it/paolo.brandimarte

Versione (provvisoria): 26 aprile 2025

NOTA: Per uso didattico interno, laurea magistrale in ingegneria matematica. Non postare e non redistribuire.

Riferimenti

- P. Brandimarte, A. Villa. *Advanced models for manufacturing systems management*. CRC Press, 1995.
- M. Pinedo. *Planning and scheduling in manufacturing and services* (2nd ed). Springer, 2009. <https://web-static.stern.nyu.edu/om/faculty/pinedo/book3/index.html>
- M. Pinedo. *Scheduling: Theory, algorithms, and systems* (6th ed). Springer, 2022.

Modelli classici di machine scheduling

Risolvere un problema di schedulazione richiede di assegnare risorse a job da eseguire nel tempo, rispettando vincoli tecnologici e di capacità, in modo da ottimizzare una misura di prestazione.

Un problema di scheduling si pone a un livello di dettaglio superiore rispetto a un problema di pianificazione della produzione (es., a livello MRP).

Esiste una grande varietà di problemi di scheduling, ma iniziamo con la definizione di un problema minimale.

I job sono caratterizzati da:

- Set di operazioni da eseguire, legate da vincoli di precedenza.
- Tempi di lavorazione per l'esecuzione su ogni macchina.
- Date di rilascio e di consegna.

Assunzioni comuni sono che un job possa essere in lavorazione su al più una macchina per volta, che ogni macchina possa lavorare un job per volta, e che le lavorazioni non possono essere interrotte.

Ognuna di tali assunzioni può essere violata: lot streaming, processori batch, pre-emption.

Consideriamo i tre job seguenti, caratterizzati dai cicli di lavorazione, da lavorare su tre macchine.

JOB	CICLO
J_1	$(M_1, 10), (M_2, 5), (M_3, 6)$
J_2	$(M_2, 5), (M_1, 8)$
J_3	$(M_1, 2), (M_3, 10), (M_2, 4)$

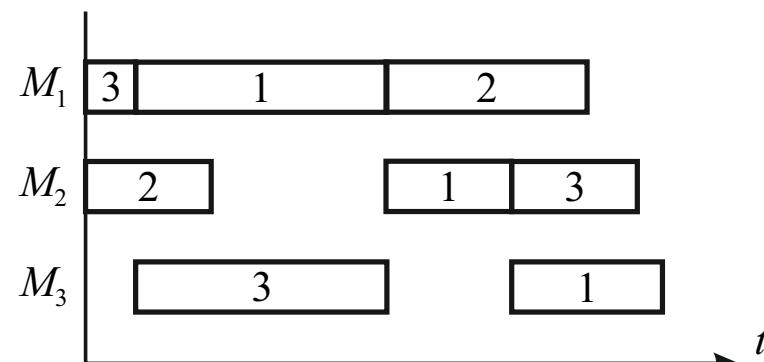
Una possibile soluzione, caratterizzata dalle sequenze

$$M_1 : J_3, J_1, J_2$$

$$M_2 : J_2, J_1, J_3$$

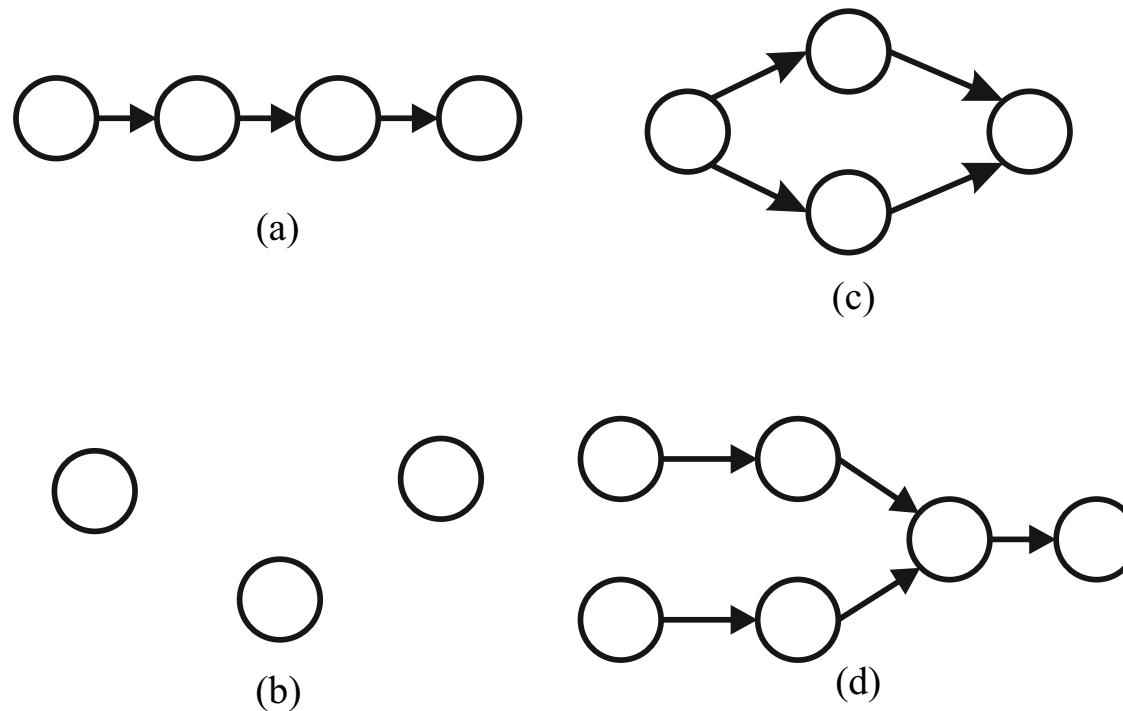
$$M_3 : J_3, J_1.$$

può essere visualizzata mediante un diagramma di Gantt.



Tipi di flusso

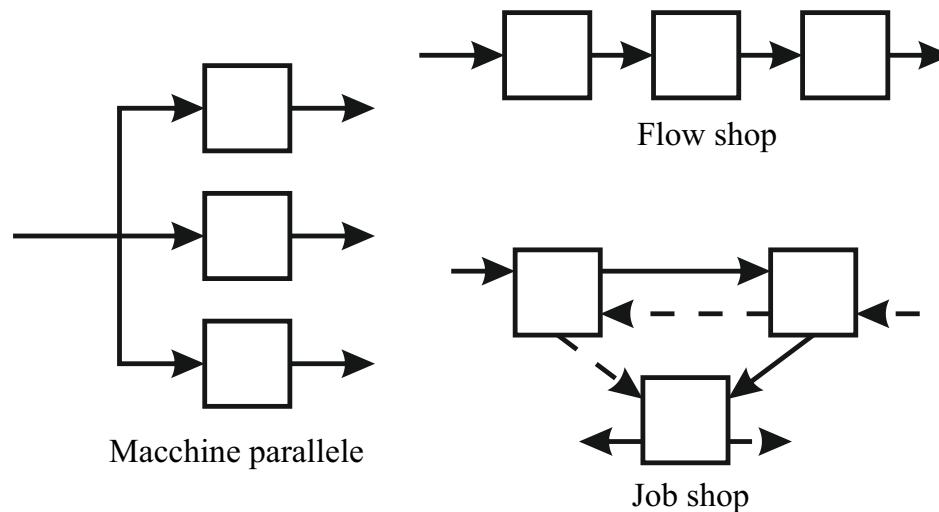
Esistono diverse strutture di precedenza tra le operazioni dei job, che caratterizzano la tecnologia del ciclo di lavorazione. Sono possibili strutture lineari, oppure strutture arbitrarie rappresentabili mediante grafi.



Le precedenze si traducono in vincoli di un problema di ottimizzazione.

Il flusso dei materiali caratterizza diverse strutture in termini di macchine:

- Macchina singola
- Macchine parallele (identiche: $p_{im} = p_i$, correlate: $p_{im} = p_i s_m$, scorrelate)
- Flow shop
- Job Shop
- Open shop



Oltre a tali strutture prototipali, si osservano varianti sul tema, e sono possibili strutture ibride (es., flexible flow shop).

Misure di prestazione

Indichiamo con C_i il tempo di completamento del job $i \in [n]$ (ovvero il tempo di completamento della sua ultima operazione).

Un tipico modo per costruire misure di prestazione è associare ad ogni job J_i un termine di penalità $\gamma_i(C_i)$.

I termini tipicamente utilizzati sono:

tempo di completamento: $\gamma_i(C_i) = C_i$;

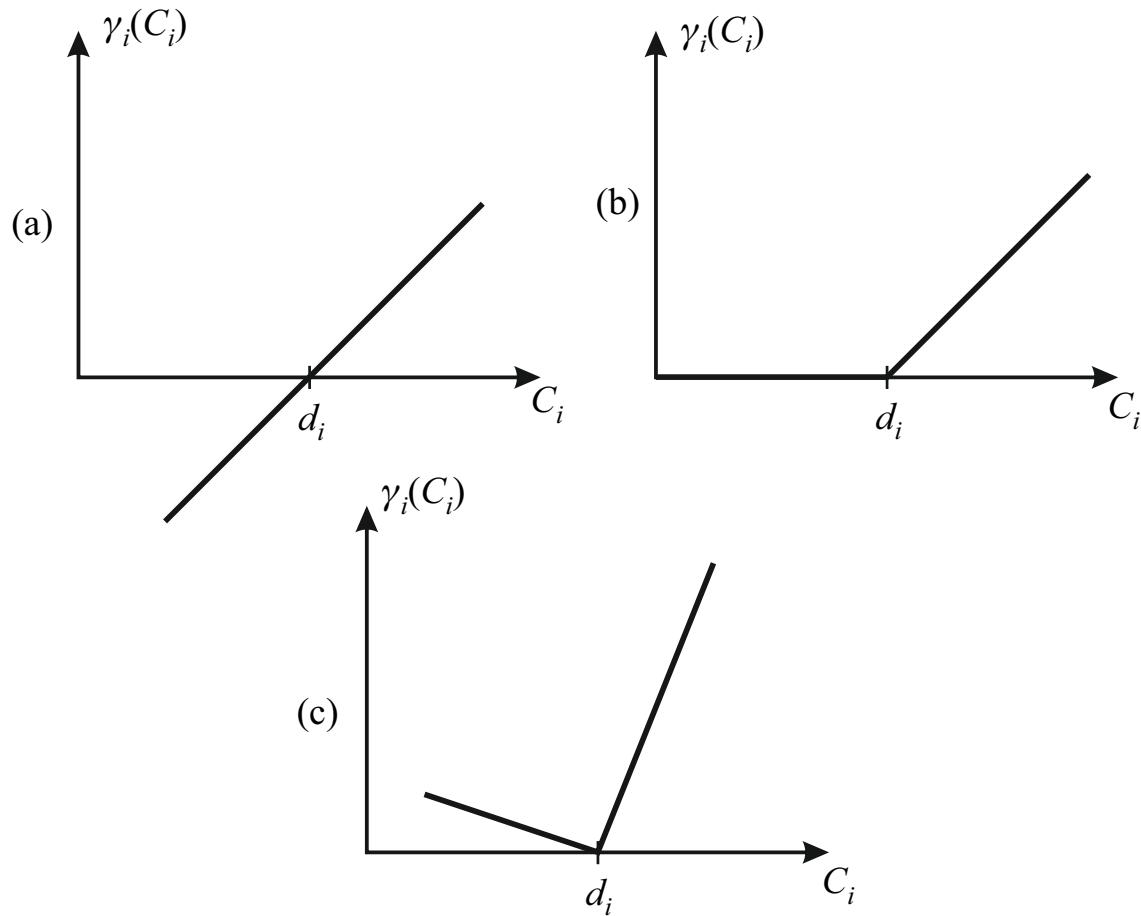
flow time: $\gamma_i(C_i) = F_i = C_i - r_i$;

lateness: $\gamma_i(C_i) = L_i = C_i - d_i$;

tardiness: $\gamma_i(C_i) = T_i = \max\{C_i - d_i, 0\}$;

earliness: $\gamma_i(C_i) = E_i = \max\{d_i - C_i, 0\}$;

indicatore di ritardo: $\gamma_i(C_i) = U_i = \begin{cases} 1 & \text{se } C_i > d_i \\ 0 & \text{se } C_i \leq d_i. \end{cases}$



È possibile associare pesi w_i ai job, allo scopo di esprimere una priorità. Si può per esempio valutare per ogni job J_i la tardiness pesata $w_i T_i$.

Sulla base di questi “mattoncini” si possono costruire misure di prestazione aggregando i termini corrispondenti ai vari job, ottenendo funzioni del tipo *minsum*, $\sum_{i=1}^n \gamma_i(C_i)$, o *minmax*, $\max_{i=1,\dots,n} \gamma_i(C_i)$.

flow time totale: $\sum_i F_i$;

flow time totale pesato: $\sum_i w_i F_i$;

massima lateness: $L_{\max} = \max_i L_i$;

tardiness totale pesata: $\sum_i w_i T_i$;

makespan: $C_{\max} = \max_i C_i$;

numero di job in ritardo: $\sum_i U_i$.

Alcune misure sono tra di loro equivalenti, nel senso che una soluzione ottima rispetto ad una di esse è ottima anche per l'altra.

Ad esempio, la lateness totale è equivalente al flow time totale.

$$\sum_{i=1}^n L_i = \sum_{i=1}^n C_i - \sum_{i=1}^n d_i = \sum_{i=1}^n F_i + \sum_{i=1}^n (r_i - d_i).$$

Le due misure di prestazione differiscono per una costante.

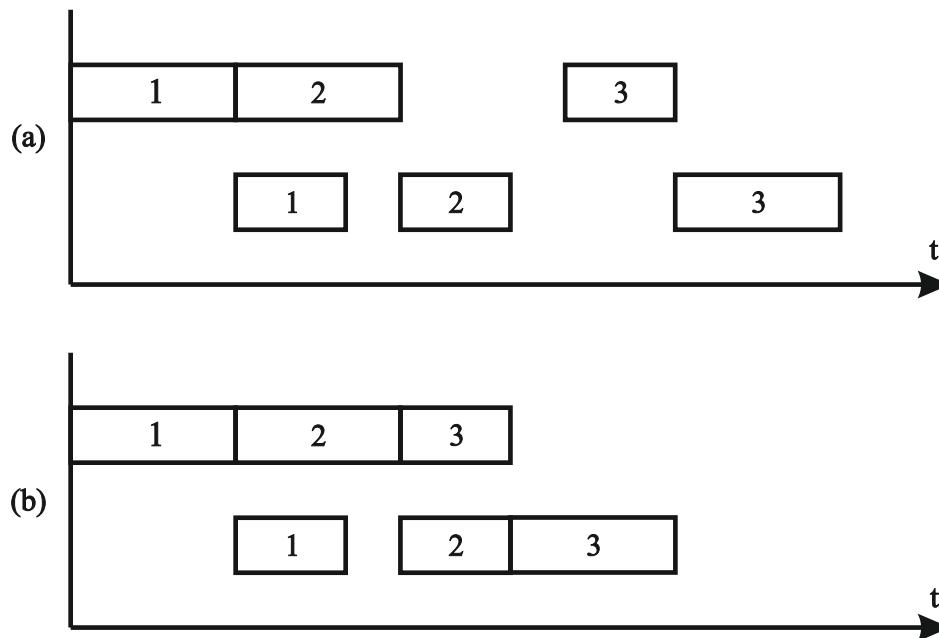
Inoltre una soluzione ottimale rispetto a L_{\max} è ottimale anche rispetto a T_{\max} (ma non è assicurato il viceversa). Infatti abbiamo

$$\begin{aligned} T_{\max} &= \max\{\max\{L_1, 0\}, \dots, \max\{L_n, 0\}\} \\ &= \max\{L_1, \dots, L_n, 0\} = \max\{L_{\max}, 0\}. \end{aligned}$$

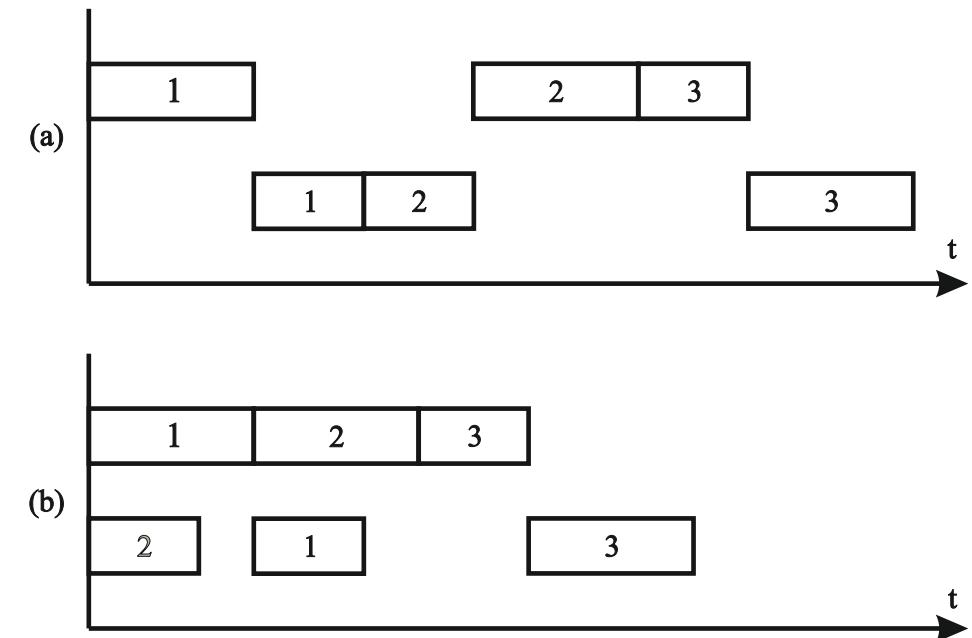
Le misure di prestazione finora elencate sono funzioni non decrescenti dei tempi di completamento; si parla in questo caso di misure di prestazione **regolari**.

Nel caso di misure di prestazione regolari si possono ricavare le informazioni di tempistica a partire dal sequenziamento, in quanto ogni operazione è eseguita al più presto (soluzioni attive).

Schedulazione non semiattiva (a)
e semiattiva (b)



Schedulazione semiattiva (a)
e attiva (b)



Nel caso di misure non regolari, ciò non è più vero, e il problema risulta molto più complesso.

Le misure regolari sono utili nel caso in cui si voglia tenere conto dell'inopportunità di completare un job prima della data di consegna; si può ad esempio adottare come misura di prestazione la somma pesata di earliness e tardiness

$$\sum_{i=1}^n (w_i^T T_i + w_i^E E_i).$$

In questo caso si ha una regione in cui tale funzione decresce al crescere del tempo di completamento del job.

In questo caso sequencing e timing sono due aspetti non coincidenti, e la soluzione ottima non è necessariamente attiva o semiattiva.

Notazione di Graham

Per classificare i problemi di schedulazione useremo una codifica a tre campi $\alpha/\beta/\gamma$ dovuta a Graham.

Il campo α descrive il layout delle macchine. Ad esempio:

- 1 indica macchina singola;
- P , Q e R indicano, rispettivamente macchine parallele identiche, correlate e scorrelate.
- F indica un flow shop ($F2$ indica un flow shop di due macchine);
- J indica un job shop.

Il campo γ descrive la misura di prestazione.

Il campo β descrive la presenza eventuale di vincoli aggiuntivi. A titolo di esempio:

- res indica la presenza di risorse aggiuntive oltre alle macchine;
- prec indica la presenza di vincoli di precedenza tra job (un vincolo di precedenza tra due job J_i e J_k *non* significa che vanno sequenziati consecutivamente; è ammesso che vi siano altri job in sequenza tra J_i e J_k);
- r_i indica che i tempi di rilascio non sono tutti nulli;
- \bar{d}_i indica la presenza di deadline hard;
- s_{ij} indica la presenza di tempi di tempi/costi di setup dipendenti dalla sequenza;
- perm indica un flow shop a permutazione (la sequenza di job si mantiene su tutte le macchine);
- no-wait indica che i job non possono attendere in un buffer tra due operazioni consecutive (ragioni tecnologiche impongono che le operazioni vengano eseguite consecutivamente senza interruzione).

Algoritmi di soluzione

Esistono alcuni casi semplici che si risolvono mediante algoritmi polinomiali:

- $1//L_{\max}$ mediante la regola EDD (earliest due date) che ordina i job in ordine crescente di due date.
- $1//\sum_i w_i p_i$ mediante la regola WSPT (weighted shortest processing time) che ordina i job in ordine decrescente del rapporto w_i/p_i .
- $F2//C_{\max}$ mediante l'algoritmo di Johnson (in questo caso la soluzione ottima applica la stessa sequenza sulle due macchine).

In generale, i problemi di schedulazione sono NP –hard e vengono risolti mediante euristiche, che possono essere semplici approcci costruttivi (one shot) o sofisticate strategie iterative (metaeuristiche basate su ricerca locale).

L'applicazione si regole di priorità fornisce una semplice strategia costruttiva (necessaria per inizializzare strategie iterative).

Esistono anche regole non banali come la regola ATC (Apparent Tardiness Cost) proposta per la soluzione di problemi $J//\sum_i w_i T_i$.

L'espressione della priorità del job J_i sulla macchina M_j al tempo t è

$$\frac{w_i}{p_{ij}} \exp \left(- \left[\frac{d_i - t - p_{ij} - \sum_{q=j+1}^{m_i} (W_{iq} + p_{iq})}{k\bar{p}} \right]^+ \right),$$

dove w_i è il peso del job J_i e p_{ij} è il suo tempo di processo sulla macchina M_j ; la notazione $[x]^+$ è equivalente a $\max\{0, x\}$; W_{iq} è una stima del tempo di attesa di J_i sulla macchina M_q ; m_i è il numero di macchine che il job deve ancora visitare; k è un parametro da selezionare; \bar{p} è il tempo di processo medio dei job in coda. L'espressione

$$t^* = d_i - p_{ij} - \sum_{q=j+1}^{m_i} (W_{iq} + p_{iq}) \quad (1)$$

può essere interpretata come il massimo tempo di inizio di J_i su M_j , ottenuto sottraendo un lead time stimato dalla due date d_i .

Se $t \leq t^*$ si può ritenere di essere in tempo, altrimenti si corre il rischio di arrivare in ritardo.

Se siamo in tempo, il termine tra parentesi quadre è positivo, e la priorità cresce esponenzialmente al crescere di t .

Quando si è in ritardo, il termine tra parentesi quadre è negativo, per cui l'argomento dell'esponenziale è zero. La priorità è quindi w_i/p_{ij} , che coincide con la regola WSPT.

Se molti job sono in ritardo la tardiness totale pesata diventa una funzione obiettivo quasi equivalente al tempo di completamento totale pesato:

$$\sum_i w_i T_i = \sum_i w_i \max\{C_i - d_i, 0\} \approx \sum_i w_i C_i - \sum_i w_i d_i.$$

Ma il problema di minimizzazione di $\sum_i w_i C_i$ è proprio risolto, su macchina singola, dalla regola WSPT.

Si può ridurre la miopia delle regole di priorità introducendo un certo grado di lookahead, ad esempio mediante strategie beam search.

Una vasta classe di algoritmi iterativi si basa su perturbazioni locali della soluzione corrente, che definiscono una struttura di vicinato.

Problemi:

- come sfuggire a minimi locali (es., tabu search o algoritmi genetici; questi ultimi richiedono attenta codifica e decodifica delle soluzioni);
- come esplorare grandi vicinati in maniera efficiente (LNS, Large Neighborhood Search basata su metaeuristiche, utile anche per misure non regolari);
- evitare la creazione di cicli.

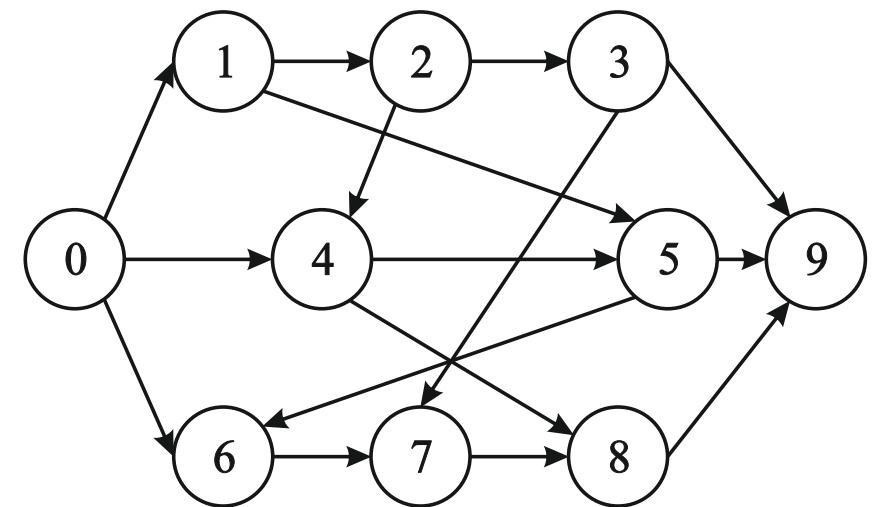
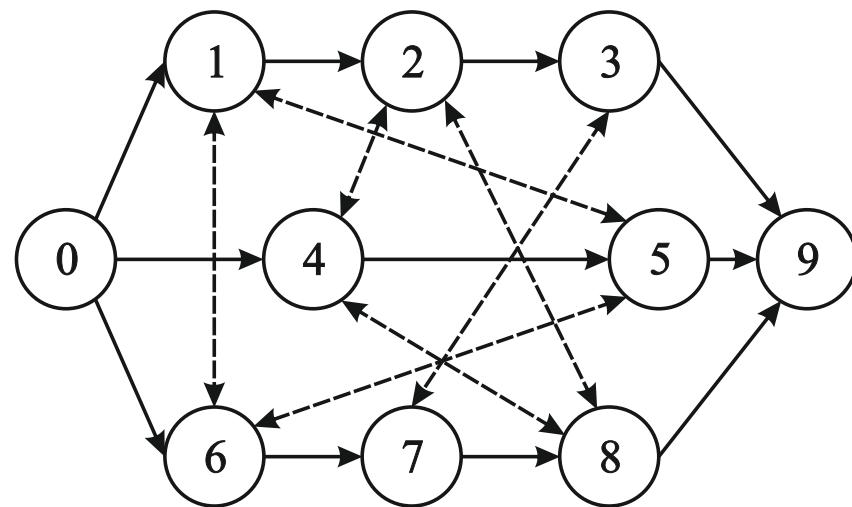
Molte di tali questioni richiedono la formulazione mediante un grafo disgiuntivo.

Grafi disgiuntivi

In un grafo disgiuntivo, i nodi rappresentano le operazioni (con l'aggiunta di nodi dummy iniziali e finali).

Gli archi congiuntivi rappresentano vincoli di precedenza tecnologici, associati ai cicli di ciascun job.

Gli archi disgiuntivi vanno orientati e rappresentano i vincoli di capacità per ogni macchina (una clique per macchina).



Il cammino critico (a lunghezza massima) da nodo iniziale a finale fornisce il make-span.

Sulla base del grafo disgiuntivo possiamo costruire un modello MILP per il problema $J//C_{\max}$. Indichiamo con $\mathcal{N} = \{0, 1, \dots, N-1, N\}$ l'insieme dei nodi, dove 0 e N sono nodi dummy; inoltre definiamo l'insieme \mathcal{P} di archi congiuntivi e l'insieme \mathcal{D} di archi disgiuntivi.

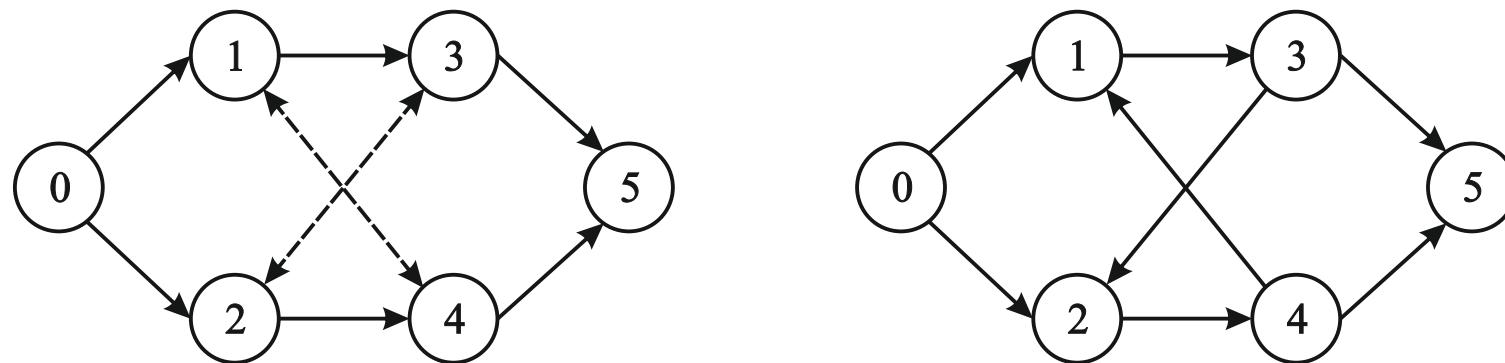
La variabile binaria x_{ij} , se posta a 1, indica che l'operazione i precede l'operazione k .

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

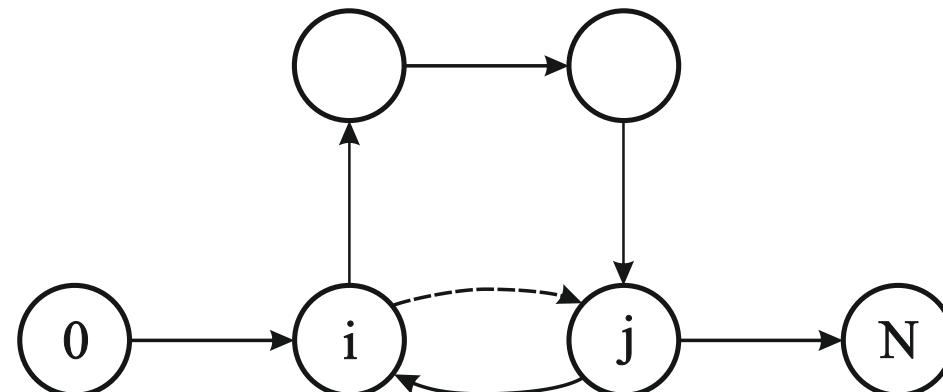
Il modello può essere facilmente adattato ad altre misure di prestazione, ma non è trattabile in pratica (bound deboli). Tuttavia esso può essere una base utile per mateuristiche e approcci LNS che risolvono parte del modello (strategie destroy and repair).

Il ruolo del cammino critico

In una strategia di ricerca locale, se si perturbano sequenze in modo arbitrario, si possono creare cicli.



Tuttavia, con riferimento a un problema $J//C_{\max}$, se si perturbano archi disgiuntivi sul cammino critico, non si possono creare cicli (e comunque sono le sole perturbazioni utili).

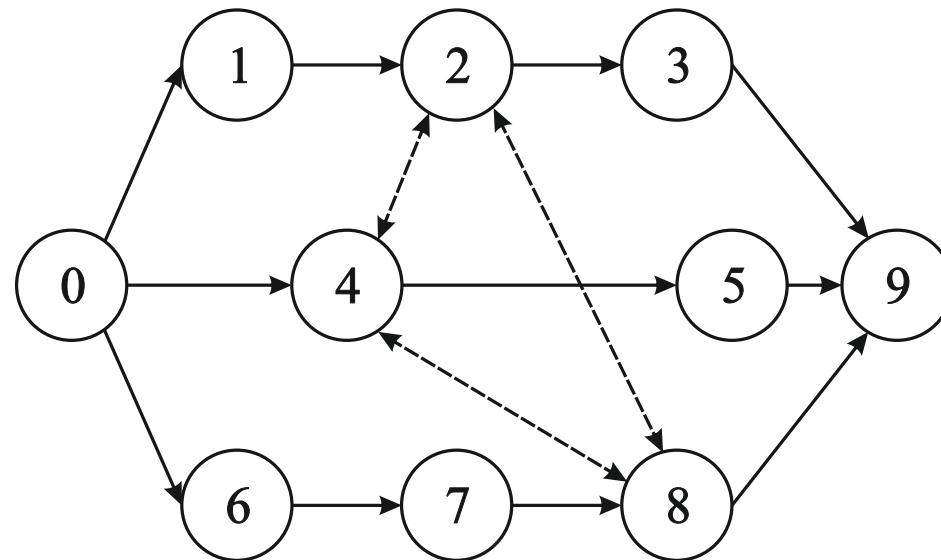


Procedura shifting bottleneck

La procedura shifting bottleneck, originariamente pensata per il problema $J//C_{\max}$, sfrutta il grafo disgiuntivo per decomporlo in un insieme di problemi del tipo $1/r_i/L_{\max}$, per i quali sono disponibili algoritmi B&B efficienti.

L'idea è che se una macchina, indicata co,e M_b , è il collo di bottiglia, potremmo pensare di trattare le altre macchine come risorse a capacità infinita.

Questo significa eliminare gli archi disgiuntivi corrispondenti alle altre macchine e mantenere solo quelli associati al collo di bottiglia.



La schedulazione del collo di bottiglia è affrontabile come un problema $1/r_i/L_{\max}$.

Consideriamo il nodo O_{ib} del grafo, corrispondente all'operazione del job J_i sulla macchina M_b , e sia C_{ib} il suo tempo di completamento.

Consideriamo il cammino critico dal nodo fittizio iniziale al nodo O_{ib} . La sua lunghezza h_{ib} è semplicemente la somma dei tempi di processo delle operazioni di J_i prima dell'operazione O_{ib} , e ne costituisce la “testa”, ovvero il tempo di rilascio dell'operazione O_{ib} .

Definiamo in modo simile la “coda” t_{ib} come lunghezza del cammino critico dal nodo O_{ib} al nodo fittizio finale. In questo caso t_{ib} è la somma dei tempi di processo delle operazioni che seguono O_{ib} .

Definiamo una data di consegna per il job J_i sulla macchina M_b , approssimando C_{\max} come

$$C_{\max} = \max_i \{C_i\} \approx \max_i \{C_{ib} + t_{ib}\}.$$

Se sottraiamo una costante arbitraria K dalla misura di prestazione, non cambiamo la soluzione ottima del problema, ottenendo quindi il problema equivalente

$$\max_i \{C_{ib} + t_{ib}\} - K = \max_i \{C_{ib} - (K - t_{ib})\} = \max_i \{C_{ib} - d_{ib}\},$$

in cui abbiamo introdotto una due date locale $d_{ib} = K - t_{ib}$.

Più lunga è la coda t_{ib} , più stretta è la data di consegna d_{ib} .

Il problema $J//C_{\max}$ si riconduce quindi ad un problema $1/r_i/L_{\max}$, in cui i tempi di rilascio sono dati dalle teste ($r_{ib} = h_{ib}$) e le date di consegna dipendono dalle code ($d_{ib} = K - t_{ib}$).

Un modo ragionevole per identificare il collo di bottiglia è risolvere i problemi $1/r_i/L_{\max}$ per tutte le macchine, definendo per ogni problema le teste e le code delle operazioni. Il collo di bottiglia sarà la macchina che ha il valore di L_{\max} peggiore.

Dopo avere risolto i problemi su macchina singola, sappiamo qual è il collo di bottiglia M_b , e abbiamo anche una sequenza per M_b . Possiamo orientare gli archi disgiuntivi corrispondenti a M_b e rischedulare le altre macchine. In pratica, si risolvono i problemi su macchina singola per le rimanenti $M-1$ macchine, avendo aggiornato teste e code.

Si procede fino ad avere schedulato tutte le macchine, congelandone una per volta. Se possono applicare raffinamenti iterativi, e la strategia si estende facilmente a a $J/r_i/L_{\max}$ (sono state proposte altre estensioni).