Indice

Scheduling: modelli, algoritmi e simulazione

Alessio Puppato, Barbara Fuoco, Andrea Rossi, Michele Lanzetta (lanzetta@unipi.it)
Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione
Università di Pisa

Sommario

A partire da una classificazione dell'architettura delle linee produttive, viene proposto un caso applicativo per mostrare le potenzialità degli algoritmi di simulazione delle sequenze di operazioni su un determinato parco macchine

INTRODUZIONE

Le attuali tendenze del mercato, la varietà della domanda dei consumatori, il breve ciclo di vita del prodotto e la pressione della concorrenza hanno spinto le aziende ad affrontare il problema della riduzione dei costi di produzione attraverso una migliore gestione delle risorse disponibili. Uno strumento fondamentale è rappresentato dagli algoritmi di scheduling per l'ottimizzazione della produzione.

Lo **scheduler** è un componente fondamentale dei sistemi operativi multitasking cioè quelli in grado di eseguire più processi di scheduling (*task*) concorrentemente. Lo scheduler si occupa di fare avanzare un processo di scheduling interrompendone, temporaneamente, un altro, realizzando così un cambiamento di contesto (context switch). Nel caso di unico processore presente nel sistema, può essere eseguito un solo processo alla volta. Un processo può essere scomposto in più processi per permettere un avanzamento dei processi in parallelo; in tal caso occorre uno scheduler per permettere la convivenza di più task sullo stesso processore. Esistono vari algoritmi di scheduling che permettono di scegliere nella maniera più efficiente possibile quale task far proseguire: ad esempio il kernel linux nella versione 2.4 ha uno scheduler di complessità O(n), mentre dalla versione 2.6 ha un algoritmo O(1), ossia in grado di determinare in un tempo costante quale processo debba essere eseguito, indipendentemente dal numero di processi in attesa [1].

I principali modelli di schedulazione sono:

1. **Macchina Singola**
2. **Macchine Parallele**
3. **Job Shop**
4. **Flexible Job Shop**
5. **Flow Shop (modello di Johnson [2])**
6. **Flexible Flow Shop**
7. **Open Shop**

Dopo un breve inquadramento teorico, nell'articolo saranno presentati due casi applicativi relativi ai punti 3. e 6.

I problemi di scheduling sono problemi decisionali in cui riveste importanza fondamentale il fattore tempo, inteso come risorsa (scarsa) da allocare in modo ottimale. In particolare, nei problemi di *scheduling su macchine* la risorsa tempo è identificata con il tempo necessario per l'esecuzione di determinate attività/operazioni (job) sulle macchine stesse.

Va sottolineato che tali problemi appartengono alla famiglia Constraint Optimisation Problem (COP) [3].

Nel contesto della fabbricazione industriale, lo scheduling consiste nel determinare l'allocazione sequenziale dei job alle macchine che ottimizza una certa funzione obiettivo, e che rispetta allo stesso tempo i vincoli imposti.

Questo problema è stato oggetto di numerosi studi nell'ambito della Ricerca Operativa, che hanno consentito di sviluppare diversi modelli che lo rappresentano e differenti metodi per risolverlo.

La scelta del processo successivo migliore compiuta senza riguardo alla minimizzazione di una funzione obiettivo viene detta "miope" perché porta ad allungare i tempi di lavoro. Per quanto risulti strano, non si può dire quale sarà il prossimo job da eseguire in una data macchina, se prima non è stato determinato lo schedule ottimo. Da qui segue la complessità cosidetta NP-*hard* (dove NP sta per *non deterministic polynomial*) dello scheduling. Tuttavia, data la natura NP-*hard* del problema, a tutt'oggi non è possibile identificare un approccio preferibile in assoluto per la sua soluzione. Tale complessità computazionale ha indirizzato la ricerca verso lo sviluppo di algoritmi euristici, che hanno come obiettivo quello di calcolare una soluzione approssimata, quanto più possibile vicina a quella ottimale [4].

Tra i vari approcci euristici proposti in letteratura i più promettenti risultano quelli che impiegano *metaeuristiche*, quali la *tabu search*, il *simulated annealing* e gli *algoritmi genetici*, che in termini di qualità della soluzione e tempi computazionali hanno dato finora risultati soddisfacenti [5].
Recentemente, è stato proposto da *Bertsekas* un nuovo approccio euristico per la soluzione di problemi di ottimizzazione combinatoria, noto come algoritmo di rollout [6].

FORMALIZZAZIONE DEL PROBLEMA DI SCHEDULING

I problemi di scheduling consistono nel determinare l'allocazione ottimale delle attività che devono essere svolte con le risorse disponibili.
Con il termine *macchina* si indica una qualsiasi risorsa come ad esempio: una macchina utensile, un centro di lavoro, un reparto, un operatore, etc. Le attività, che possono consistere in una sola o più operazioni elementari (ciclo di lavorazione), come ad esempio sfacciatura, cilindratura, misura o montaggio di un particolare meccanico, sono indicate con il termine *job*.
Soventemente però le attività sono definite con la parola *task*, ed un job è a quel punto inteso come un insieme di task tecnologicamente legate fra loro, che devono essere eseguite affinchè il job sia ultimato.
A ciascun job possono essere associate molteplici informazioni [7]:

- ✓ *Tempo di processamento (processing time)* $\boldsymbol{\tau_{ij}}$: rappresenta il tempo necessario al completamento del job *j*-esimo sulla *i*-esima macchina. Se il tempo di processamento è indipendente dalla macchina si indica semplicemente con $\boldsymbol{\tau_j}$.
- ✓ *Tempo di rilascio* (*release date*) $\boldsymbol{r_j}$: indica l'istante di tempo (rispetto a un tempo iniziale) prima del quale non è possibile iniziare l'esecuzione del job *j*-esimo. Può rappresentare l'istante in cui diventano disponibili le materie prime o i semilavorati per la lavorazione di un certo prodotto.
- ✓ *Tempo di consegna* (*due date*) $\boldsymbol{d_j}$: indica l'istante di tempo (rispetto a un tempo iniziale) entro il quale l'esecuzione del *j*-esimo job dovrebbe essere terminata. In genere, la violazione di un tempo di consegna comporta dei costi, delle penalità che si applicano sul ricavo pattuito con l'ordinativo (o commessa d'ordine). Nel caso in cui la *due date* debba essere assolutamente rispettata essa prende il nome di *deadline*.
- ✓ *Peso* $\boldsymbol{w_j}$: tale quantità rappresenta l'importanza relativa del job *j*-esimo rispetto agli altri; può rappresentare, ad esempio, il costo di mantenimento del job nel sistema (ad es. costo di immagazzinamento). E' dunque un indice di priorità del job *j*-esimo rispetto agli altri.

Consideriamo ora le caratteristiche relative all'ambiente produttivo. A livello industriale esiste una grande varietà di layout ed architetture di impianto, le quali differiscono fra loro per il numero di macchine presenti e/o per l'ordine con cui i job sono eseguiti su queste.
I sistemi più semplici sono i seguenti[8]:

1. *Macchina Singola*: in questo sistema tutti i job richiedono la medesima risorsa produttiva per poter essere completati (Figura 1). Tale caso è ampiamente trattato in letteratura vista la sua semplicità.

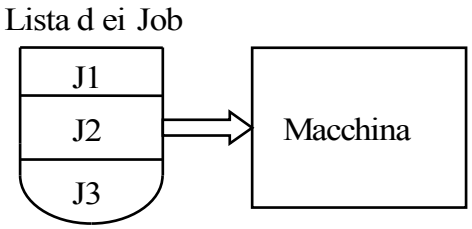


Figura 1: Sistema produttivo a Macchina Singola

2. *Macchine Parallele.* Si possono avere tre tipologie di architettura, schematizzate in Figura 2.
   - *Macchine parallele identiche*: in questo caso i prodotti possono essere indifferentemente lavorati su m macchine identiche.
   - *Macchine parallele generiche*: in questo caso ci sono ancora m macchine come nel caso precedente ma i tempi di lavorazione $\boldsymbol{\tau_{ij}}$ dipendono oltre che dal job j anche dalla macchina i su cui il lavoro viene eseguito. Ogni macchina è caratterizzata da una sua velocità v_i.
   - *Macchine parallele scorrelate*: il sistema produttivo è una generalizzazione del caso precedente. La macchina i può eseguire il job j con una velocità v_{ij}. Se le velocità delle macchine fossero indipendenti dai job (cioè $v_{ij} = v_i$ per ogni j) si ricadrebbe nel caso precedente.

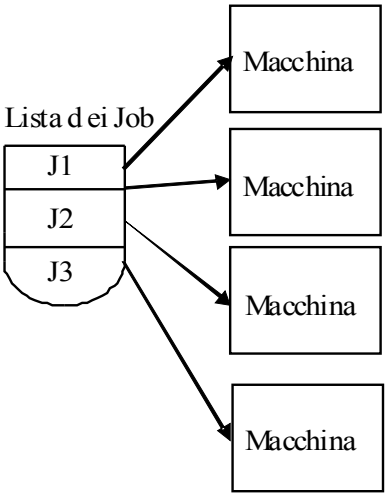


Figura 2: Sistema produttivo a Macchine Parallele

3. *Flow Shop*: in questo caso il sistema consiste di *m* macchine in serie e ciascun job deve essere eseguito da ciascuna delle macchine successivamente, ossia l'ordine in cui i job visitano le macchine è lo stesso per tutti i job (*routing* fisso). Inoltre il flusso di lavorazione è unidirezionale (Figura 3).

J2=(M1, M3, M2)
J1=(M1, M3, M2)
J3=(M1, M3, M2)

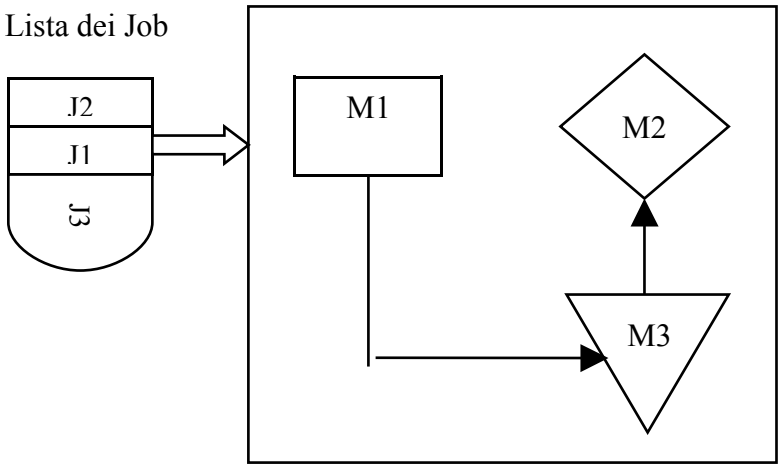


Figura 3: Sistema produttivo a Flow Shop

Da notare che l'ordine delle operazioni è fisso per ogni job ma i tempi di lavorazione su ogni macchina τ_{ij} possono essere differenti da job a job.

4. *Flexible Flow Shop*: è una generalizzazione dei sistemi produttivi a Flow Shop e di Macchine Parallele. Invece di avere *m* macchine in serie, vi sono *c* stadi di produzione in serie, ognuno con un certo numero di identiche macchine in parallelo. Ogni job deve essere lavorato prima allo

stadio 1, poi allo stadio 2 e così via, ed in ogni stadio il job può essere lavorato su una qualunque delle macchine di quello stadio.

5. *Job Shop*: anche in questa architettura vi sono *m* macchine, ma ciascun job ha un proprio ordine con cui visitarle. Si noti che il Job Shop è una generalizzazione del Flow Shop in cui il flusso delle lavorazioni non è unidirezionale.

J1=(M1, M3, M2)
J2=(M1, M2, M3)
J3=(M2, M3, M1)

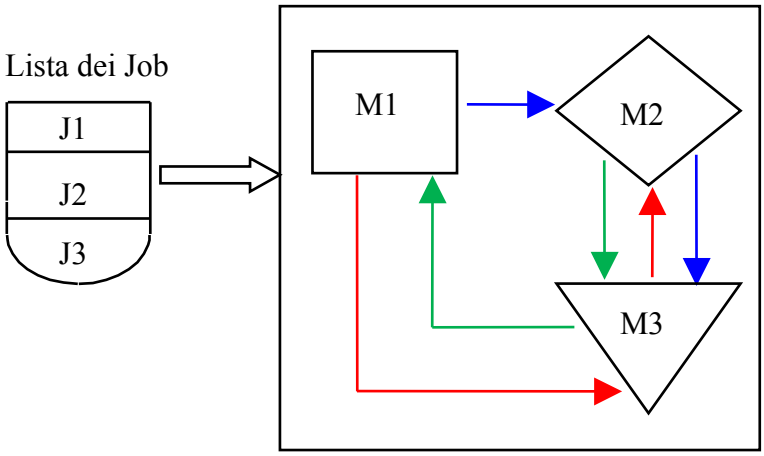


Figura 4: Esempio di sistema produttivo a Job Shop

Come si può vedere dall'esempio in Figura 4, l'ordine delle operazioni (*routing*) per ciascun job è fisso ed è diverso da job a job; inoltre i tempi di lavorazione τ_{ij} su ogni macchina possono essere differenti da job a job.

6. *Flexible Job Shop:* è una generalizzazione dei sistemi produttivi a Job Shop ed a Macchine Parallele. Invece di *m* macchine ci sono *c* workcenter (isole, aree produttive, celle) ognuno con un certo numero di identiche macchine in parallelo. Ogni job deve essere processato, seguendo un suo ordine ben preciso, nei vari workcenter e in ognuno di questi può essere lavorato su qualsiasi macchina [9].

7. *Open Shop*: in questo caso, schematizzato in Figura 5, ciascun job può essere processato sulle *m* macchine senza alcun ordine specifico (routing non fisso). La scelta del routing di ogni job può essere determinata sulla base di considerazioni logistico-gestionali non essendo imposta da ragioni tecnologiche.

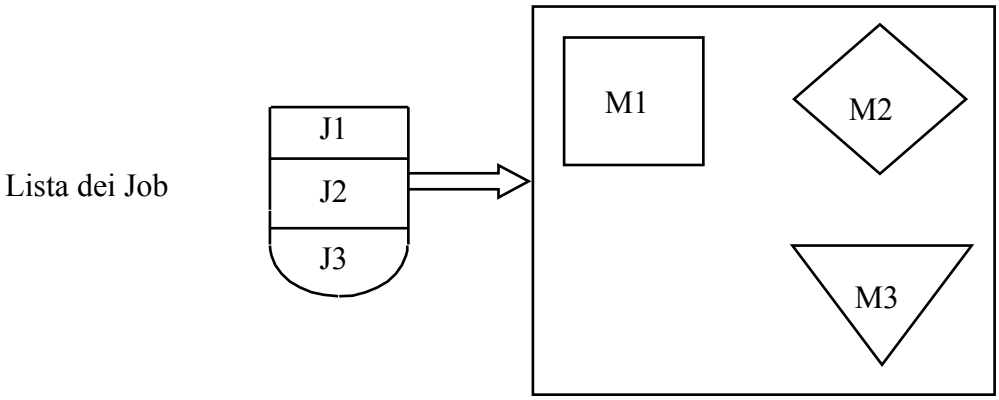


Figura 5: Sistema produttivo a Open Shop

Oltre alle caratteristiche dei job e del sistema produttivo, per definire esaurientemente un problema di scheduling occorrono ulteriori informazioni, tra le quali:

- *Tempi di setup*: indica il tempo necessario a riconfigurare la macchina nel passaggio da un job *j*-esimo ad un altro *k*-esimo e si indica con $\mathbf{s_{jk}}$. Se il setup dipende dalla precedente operazione eseguita sulla macchina si parla di *sequence dependent setup times*, mentre nel caso in cui il setup non dipenda dalla precedente operazione nel routing del job si parla di *sequence independent setup time.*
- *Preemption* (priorità o importanza): caratteristica che, se presente, indica che è consentito interrompere la realizzazione di un job (si intende che si è già iniziato a lavorare) a favore di altri con priorità più alta. I job in sospeso saranno ripresi in un secondo momento, dopo che i job con priorità maggiore sono stati ultimati.
- *Precedence constraints* (vincoli di precedenza): significa che esistono delle relazioni di precedenza tra i vari job; può accadere infatti che un job debba aspettare il completamento degli altri prima di essere eseguito.
- *Machine breakdown*: avaria di un apparato o di un dispositivo della macchina che non ne consente temporaneamente l'utilizzo. Il lasso di tempo durante il quale il macchinario non è

disponibile si assume noto (ad esempio a seguito di una manutenzione programmata).

Prima di presentare le funzioni obiettivo più usate nei problemi di scheduling, occorre introdurre i seguenti ulteriori parametri da associare ai job.

La lettera *i* viene utilizzata come pedice per riferirsi alla macchina, mentre con la lettera *j* ci si riferisce al job.

- *Tempo di completamento* $\boldsymbol{c_{ij}}$: indica il tempo (rispetto a un tempo iniziale) necessario affinchè il *j*-esimo job sia completato sulla macchina *i*-esima. Se il pedice *i* viene omesso c_j si riferisce all'intervallo di tempo necessario perchè il job *j*-esimo concluda la sua lavorazione sull'ultima macchina in cui doveva essere processato.
- *Lateness* $\boldsymbol{L_j}$: rappresenta la differenza tra il tempo di completamento e il tempo di consegna del *j*-esimo job, ovvero $L_j = c_j - d_j$. Se tale differenza è positiva, la lateness indica un ritardo, se negativa un anticipo rispetto al tempo di consegna.
- *Tardiness* $\boldsymbol{T_j}$: il tardiness di un job *j*-esimo è definito come $T_j = \max(0; c_j - d_j) = \max(0; L_j)$. Essa è quindi una quantità sempre positiva o nulla, per cui a differenza del lateness, quest'ultimo non considera un vantaggio il fatto che un job sia in anticipo rispetto al relativo tempo di consegna.
- *Earliness* $\boldsymbol{E_j}$*:* è definito come $E_j = \min(0; L_j)$. Esso è uguale al lateness se il job è in anticipo (ovvero $L_j < 0$) ed è nullo se in ritardo. Tale valore risulta di interesse in un ottica di just in time.
- *Unit penalty* $\boldsymbol{U_j}$*:* è definita come $U_j = 1$ se $C_j > d_j$ mentre $U_j = 0$ se $C_j \leq d_j$ [10].

OBIETTIVI DELLO SCHEDULING

Una soluzione di un problema di scheduling prende il nome di *schedule* o *piano di produzione (production plan)*. In termini generali, uno schedule è una descrizione completa dell'utilizzo temporale delle macchine da parte dei job che devono essere processati.

Spesso si distingue il concetto di *sequenza* da quello di *schedule*; la *sequenza* specifica solo l'ordine in cui i job devono essere eseguiti da ciascuna macchina, lo *schedule* ne specifica anche gli istanti d'inizio [11].

Dopo l'introduzione di queste grandezze, si possono esaminare alcuni esempi di **funzioni obiettivo**.

- Minimizzazione del *makespan*
- Minimizzazione del *numero di job in ritardo*
- Minimizzazione del *lateness*

- Minimizzazione del *tardiness*
- Minimizzazione del *flow time* o tempo di attraversamento
- Minimizzazione del *work in process WIP*, cioè del numero di job in lavorazione
- Massimizzazione della *saturazione delle macchine*, il quale coincide con la minimizzazione del makespan
- Minimizzazione del *tempo di setup complessivo*

Makespan o tempo di completamento $\boldsymbol{C_{max}}$: è equivalente al *tempo di completamento* dell'ultimo job in lavorazione ovvero $C_{max} = \max(c_j)$. Minimizzare questo valore permette di ottenere un alto *coefficiente di utilizzazione delle macchine.*

Maximum lateness $\boldsymbol{L_{max}}$: è definita come $L_{max} = \max(L_j)$, individua la massima violazione della data di consegna, cioè lo scarto maggiore tra il tempo in cui si finisce di lavorare un job e quello in cui il job dovrebbe essere già consegnato. Potrebbe anche essere negativo nel caso in cui tutti i job terminino prima del tempo di consegna previsto.

Total weighted completion time $\sum \boldsymbol{w_j C_j}$: tale quantità, nota altresì con il nome *flow time*, è definita come la somma pesata dei tempi di completamento dei job (*weighted flow time*). Essa è un indicatore del work in process aziendale.

Total weighted tardiness $\sum \boldsymbol{w_j T_j}$: funzione di costo più generica rispetto al *total weighted completion time*, rappresenta la somma pesata dei ritardi nell'esecuzione dei job rispetto alle loro date di consegna.

Weighted number of tardy jobs $\sum \boldsymbol{w_j U_j}$: rappresenta il numero di job in ritardo rispetto alla propria due date ed è spesso utilizzata nella pratica.

Total setup time $\sum \boldsymbol{SU_i}$: rappresenta il tempo di setup complessivo ed è definito come $\sum_{i=1}^{m} SU_i$, dove SU_i è il tempo di setup sulla macchina *i*-esima per lavorare l'insieme dei job assegnati.

In passato la ricerca si è concentrata soprattutto su modelli che cercavano di minimizzare funzioni costituite da un solo obiettivo, mentre attualmente si è cominciato ad analizzare e studiare anche modelli caratterizzati da funzioni multiobiettivo (*multiple objective functions*) [12].

Dalle considerazioni fin qui fatte si può concludere che un problema di scheduling è univocamente descritto da tre fattori:

1. Architettura del sistema produttivo
2. Parametri del processo ed eventuali vincoli
3. Scelta della funzione da ottimizzare

Per classificare i problemi di scheduling è spesso usata la notazione di Graham **α/β/γ** nella quale:
α = individua il sistema produttivo (layout delle macchine);
β = indica la presenza di eventuali vincoli (fornisce dei dettagli sulle caratteristiche del processo e sui vincoli che devono essere rispettati, ad esempio su release date, preemption, precedence constrains, breakdowns, sequence dependent setup times, etc.);
γ = descrive la funzione obiettivo che si vuole minimizzare (misura di prestazione) [13].

Alcuni esempi sono riportati anche in [14].
P*m* || C_{max} = *m* macchine identiche parallele, minimizzare il makespan
P*m* |prec| C_{max} = *m* macchine parallele identiche, precedenze generiche tra job, minimizzare il makespan
1 |r_j| L_{max} = Macchina Singola, release time, minimizzare la massima lateness
J*m* || C_{max} = Job Shop con *m* macchine, minimizzare il makespan

ALGORITMI DI SCHEDULING

Nelle sezioni precedenti sono state presentate politiche di scheduling che specificano gli obiettivi dello scheduler. In questa sezione vengono introdotti alcuni dei più noti *algoritmi di scheduling* che, determinano quale sarà il migliore piano di produzione (o schedule); questi algoritmi decidono quando e quanto a lungo vada eseguito un piano. L'algoritmo, che può richiedere anche ore di calcolo, determina anche quando e per quanto sia possibile interromperlo, in funzione di priorità, tempo di esecuzione, tempo che lo separa dal completamento e di altre caratteristiche del processo. Tuttavia durante tutti questi anni molti problemi di scheduling sono stati classificati, nell'ambito della teoria della complessità computazionale, nella classe dei problemi NP-*hard*. Per questi problemi, cosiddetti "difficili", sebbene la questione sia ancora aperta, sono state studiate e sviluppate diverse procedure *euristiche* che possono essere adattate a una grande varietà di problemi di scheduling ed implementate abbastanza facilmente nei diversi sistemi produttivi industriali.

Tutte queste tecniche non garantiscono di ottenere una *soluzione ottima* al problema, ma si prefiggono di trovare *buone soluzioni* in *tempi relativamente brevi*.
In particolare sono presenti due tipologie di algoritmi euristici applicabili al problema dello scheduling [15]:

- **gli euristici costruttivi** (constructive heuristic);
- **gli euristici migliorativi** (improvement heuristic).

Per quanto riguarda le procedure del primo tipo esse agiscono costruendo gradualmente una soluzione ammissibile, cercando di contenere il costo della funzione obiettivo della soluzione stessa; gli euristici migliorativi, al contrario, partono già da una soluzione, che può anche non essere ammissibile, e cercano, modificandola un po' ad ogni passo, di ottenere nuove soluzioni con valore migliore della funzione obiettivo.
Nell'ottimizzazione di una certa funzione non tutte le soluzioni fornite dall'algoritmo di risoluzione possono essere compatibili con l'impianto produttivo in esame o con il ciclo di lavoro previsto per il prodotto. Uno schedule che rispetta tutti i vincoli del problema, vincoli tecnologici di precedenza tra operazioni, l'impossibilità di interrompere un job durante il processamento etc. definiti precedentemente, si definisce *schedule ammissibile*.

ESEMPIO APPLICATIVO

I principi generali esposti in precedenza (problema di scheduling e ottimizzazione) possono essere estesi ad un particolare caso aziendale.
Nel caso esaminato, l'azienda deve produrre una commessa di 5 prodotti diversi, adottando un **sistema produttivo a Job Shop**, dove ogni manufatto ha un determinato routing tecnologico fissato da esigenze meccaniche e di corretto funzionamento dei prodotti stessi (Tabella 1 de Tabella 2).
Sulla base dei dati di input, quali i vari tempi di processamento, i tempi di set up relativi al riassetto delle macchine, la movimentazione e il trasporto (material handling) e fatte le dovute ipotesi e semplificazioni, verranno confrontati i diversi risultati ottenuti con vari algoritmi di risoluzione aventi come fine quello di **minimizzare il makespan** soddisfacente tutte le precedenze e i vincoli.
Sebbene la minimizzazione del makespan possa non essere percepita come una buona funzione obiettivo, il suo uso accademico ed industriale è largamente diffuso. Questo criterio è stato storicamente molto significativo e fu il primo obiettivo applicato dai ricercatori fin dai primi anni 50 grazie alla sua semplicità dal punto di vista matematico. In questo articolo viene usata tale

funzione obiettivo per offrire una base di partenza al lettore in vista di affrontare con più solide basi problemi pratici più complessi e di maggiore interesse applicativo.

Il layout produttivo rappresentato in è costituito da 5 isole flessibili di produzione e un sistema automatico (anelli circolari interno ed esterno) di movimentazione dei pallet contenenti i prodotti (work in process).

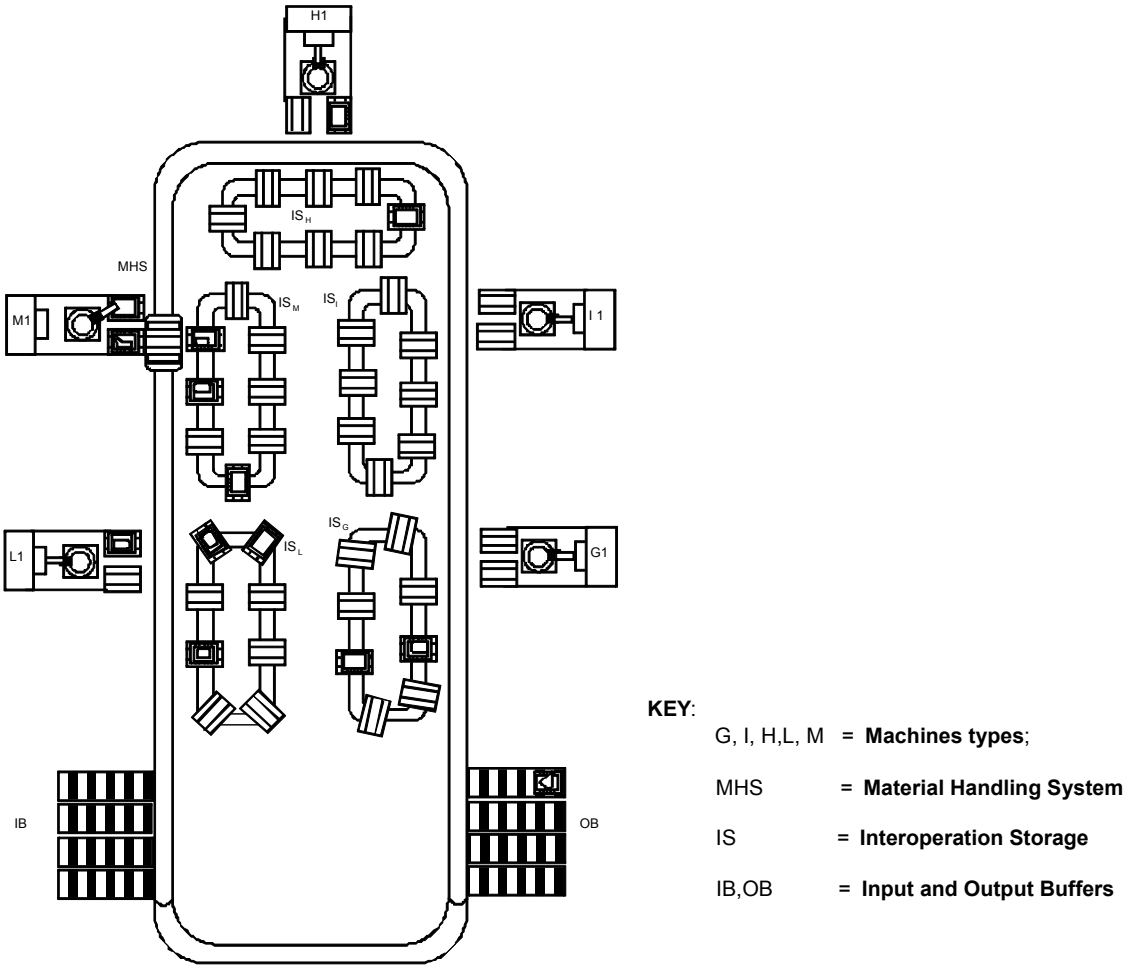


Figura 6: Esempio di layout di impianto del tipo Job Shop

Si assumono le seguenti ipotesi semplificative per il modello adottato:

- Disponibilità totale delle macchine nell'arco temporale esaminato;
- Ogni job ha lo stesso peso $w_j = 1$;
- Release date r_j e due date d_j nulli;
- Ogni job ha il proprio flusso/percorso attraverso le macchine (routing tecnologico) il quale è indipendente dagli altri job ovvero non ci sono vincoli di precedenza tra operazioni di differenti job;
- Le operazioni non possono essere interrotte (*non-preemption*);
- Ogni macchina può eseguire solo un job e ciascun job può essere lavorato solo da una macchina alla volta (*capacity constraints*).

	Jobs Type				
	J_1	J_2	J_3	J_4	J_5
Job #	J_{1-1} J_{1-2} J_{1-3}	J_{2-1} J_{2-2} J_{2-3}	J_{3-1} J_{3-2} J_{3-3}	J_{4-1} J_{4-2} J_{4-3}	J_{5-1} J_{5-2} J_{5-3}
	Machine – Setup – Time				
	4 – A - 12	2 – B - 20	1 – C - 10	5 – D - 6	3 – E - 18
	1 – A - 17	5 – B - 9	2 – C - 13	3 – D - 21	4 – E - 10
	2 – A - 7	1 – B - 16	3 – C - 15	4 – D - 11	5 – E - 22
	5 – A - 28	3 – B - 19	4 – C - 26	2 – D - 14	1 – E - 13
	3 – A - 11	4 – B - 13	5 – C - 8	1 – D - 23	2 – E - 19
	4 – A - 12	2 – B - 20	1 – C - 10	5 – D - 6	3 – E - 18

Tabella 1: Per ciascuno dei job sono riportati per colonne: i routing tecnologici, i set-up richiesti (A, B, C, D, E) ed i tempi di processamento.

Setup	A	B	C	D	E
A		1	2	3	4
B	2		4	2	3
C	1	3		4	2
D	4	2	3		1
E	3	4	1	2	

Tabella 2: Tempi di set-up necessari in funzione del set-up già presente sulla macchina (penalità).

Per la soluzione del problema viene utilizzato il software per la pianificazione automatica dei cicli *Lekin Sheduler*® [16].

In Figura 7 è mostrata l'interfaccia del programma; si riportano inoltre esempi delle finestre principali ottenute in output. Le Figure da 8 a 11 illustrano le principali funzionalità del programma.

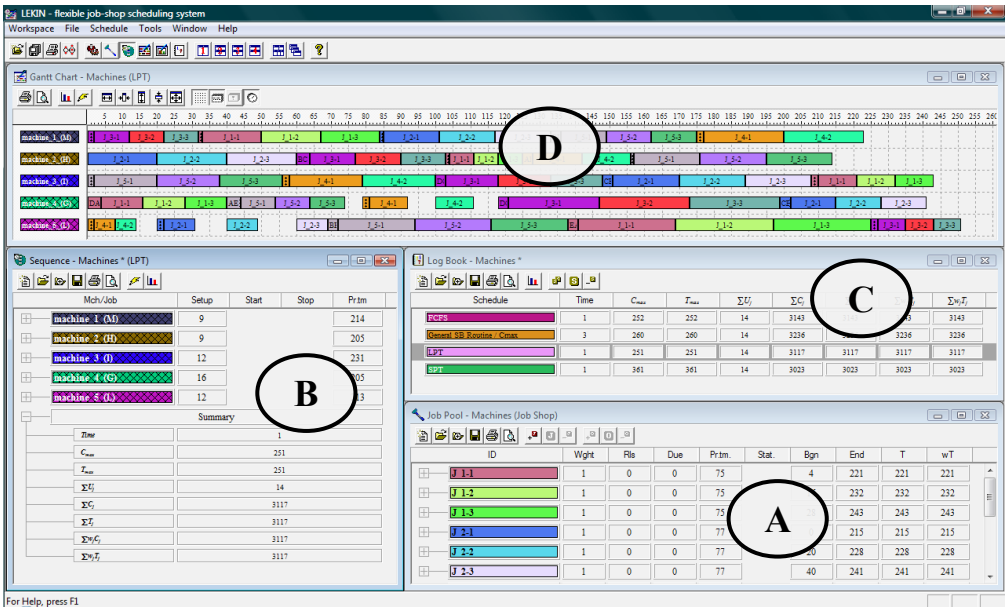


Figura 7: Schermata del software Lekin Scheduler [16] riguardante il caso esaminato, con i dati relativi al sistema produttivo (A) e al prodotto (D) inseriti dall'utente e il piano calcolato (B-C).

Il programma permette di testare diversi algoritmi di ottimizzazione al fine di determinare il minimo della funzione obiettivo, nel caso specifico il makespan.

Si riportano i seguenti algoritmi più noti [17]:

- *First-come, first-served (FCFS)*: ordinamento secondo il quale il job che arriva alla macchina per primo è il primo ad essere processato (algoritmo di tipo *FIFO first in, first out*). Essendo un algoritmo non real time, non preemptive e con prestazioni medie modeste è scarsamente utilizzato.

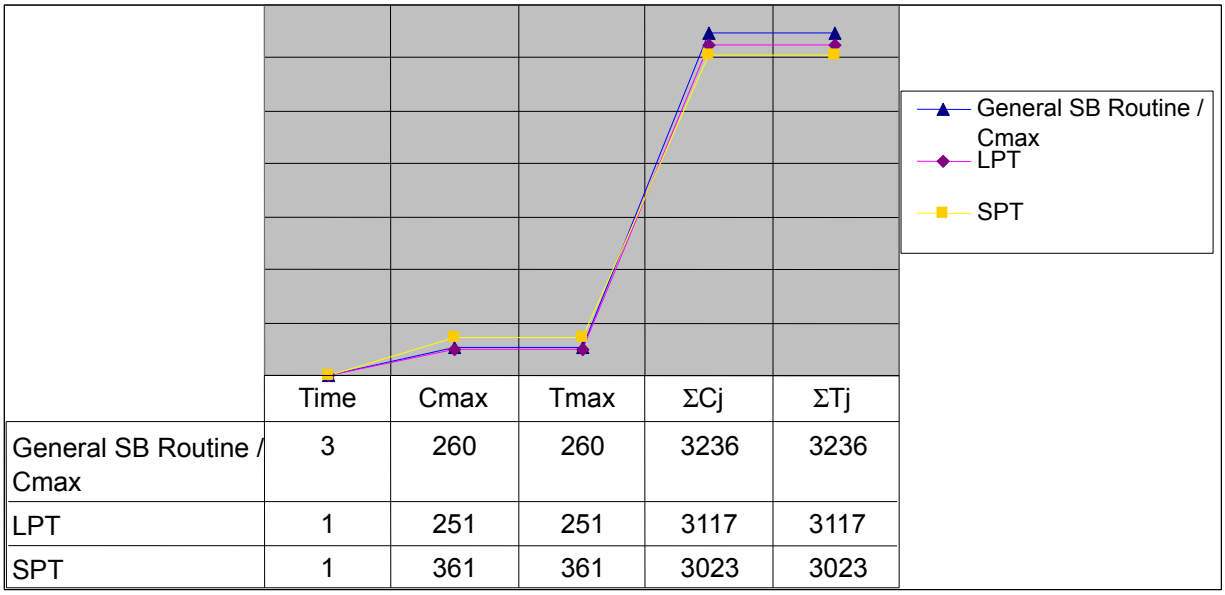
- *Shortest processing time (SPT)*: ordinamento secondo il quale il job che richiede il più basso tempo di processamento è il prossimo job ad essere processato. Si ordina per tempi di processamento crescenti.

- *Longest processing time (LPT)*: ordinamento secondo il quale il job che richiede il più alto

tempo di processamento è il prossimo job ad essere processato. Si ordina per tempi di processamento decrescenti.

- *General Shifting Bottleneck (General SB):* identifica il macchinario che influenza in modo più consistente il valore del makespan rispetto alle altre e tale macchina viene detta bottleneck ("collo di bottiglia"). L'algoritmo ad ogni passo individua la macchina che costituisce il bottleneck corrente e fissa il sequenziamento su quest'ultima. Al passo successivo, ricerca la macchina che, tra quelle ancora non sequenziate, risulta maggiormente critica, sequenzia quest'ultima e così via. Siccome il ruolo di macchinario bottleneck viene rivestito da una macchina diversa ad ogni passo, l'algoritmo viene detto shifting bottleneck. Questo algoritmo è di tipo euristico, per cui non vi è alcuna garanzia che la soluzione trovata sia quella ottima.

Gli algoritmi citati sono stati testati sul caso in esame tramite il programma descritto e i risultati sono sintetizzati nella Tabella 3



	Time	Cmax	Tmax	ΣCj	ΣTj
General SB Routine / Cmax	3	260	260	3236	3236
LPT	1	251	251	3117	3117
SPT	1	361	361	3023	3023

Tabella 3: Confronto tra i tre algoritmi di ottimizzazione testati e per righe (General shifting bottleneck, Longest processor time, Shortest processor time). La tabella riporta per colonne: il tempo di calcolo in secondi, il makespan, il tardiness, il total makespan ed il total tardiness.

Sequence - Machines (LPT)

Mch/Job	Setup	Start	Stop	Pr.tm
machine_1_(M)	9			214
J_3-1	2	2	12	10
J_3-2	0	12	22	10
J_3-3	0	22	32	10
J_1-1	1	33	50	17
J_1-2	0	50	67	17
J_1-3	0	67	84	17
J_2-1	1	85	101	16
J_2-2	0	101	117	16
J_2-3	0	117	133	16
J_5-1	3	136	149	13
J_5-2	0	149	162	13
J_5-3	0	162	175	13
J_4-1	2	177	200	23
J_4-2	0	200	223	23
machine_2_(H)	9			205
J_2-1	0	0	20	20
J_2-2	0	20	40	20
J_2-3	0	40	60	20
J_3-1	4	64	77	13
J_3-2	0	77	90	13
J_3-3	0	90	103	13
J_1-1	1	104	111	7
J_1-2	0	111	118	7
J_1-3	0	118	125	7
J_4-1	3	128	142	14
J_4-2	0	142	156	14
J_5-1	1	157	176	19
J_5-2	0	176	195	19
J_5-3	0	195	214	19
machine_3_(I)	12			231
J_5-1	2	2	20	18
J_5-2	0	20	38	18
J_5-3	0	38	56	18
J_4-1	2	58	79	21
J_4-2	0	79	100	21
J_3-1	3	103	118	15
J_3-2	0	118	133	15
J_3-3	0	133	148	15
J_2-1	3	151	170	19
J_2-2	0	170	189	19
J_2-3	0	189	208	19
J_1-1	2	210	221	11
J_1-2	0	221	232	11
J_1-3	0	232	243	11
machine_4_(G)	16			205
J_1-1	4	4	16	12
J_1-2	0	16	28	12
J_1-3	0	28	40	12
J_5-1	4	44	54	10
J_5-2	0	54	64	10
J_5-3	0	64	74	10
J_4-1	2	81	92	11
J_4-2	0	100	111	11
J_3-1	3	121	147	26
J_3-2	0	147	173	26
J_3-3	0	173	199	26
J_2-1	3	202	215	13
J_2-2	0	215	228	13
J_2-3	0	228	241	13

Sequence - Machines (LPT)

Mch/Job	Setup	Start	Stop	Pr.tm
machine_5_(L)	12			213
J_4-1	2	2	8	6
J_4-2	0	8	14	6
J_2-1	2	22	31	9
J_2-2	0	40	49	9
J_2-3	0	60	69	9
J_5-1	3	72	94	22
J_5-2	0	94	116	22
J_5-3	0	116	138	22
J_1-1	3	141	169	28
J_1-2	0	169	197	28
J_1-3	0	197	225	28
J_3-1	2	227	235	8
J_3-2	0	235	243	8
J_3-3	0	243	251	8

Summary

Time	1
C_{max}	251
T_{max}	251
$\sum U_j$	14
$\sum C_j$	3117
$\sum T_j$	3117
$\sum w_j C_j$	3117
$\sum w_j T_j$	3117

Figura 8: Dettaglio (A) della Figura 7. Output grafico ("Sequence") del software Lekin Scheduler con i tempi di setup e le date di inizio e fine processamento pianificati secondo l'algoritmo LPT utilizzato.

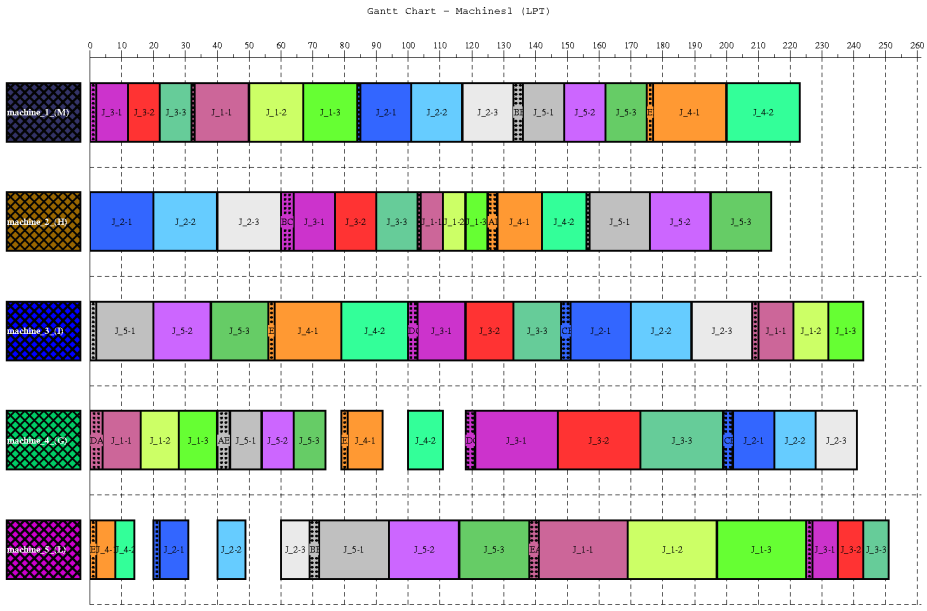
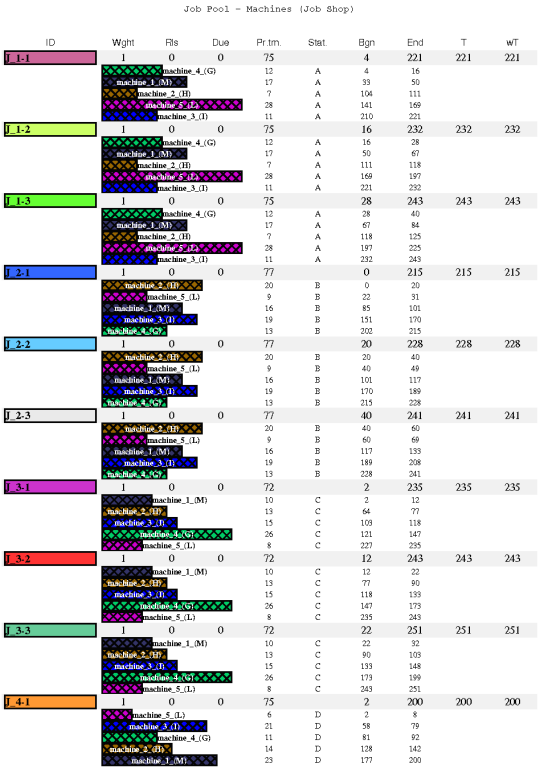


Figura 9: Dettaglio (B) della Figura 7 (diagramma di Gantt). Per ciascuna macchina è visualizzata la sequenza di lavorazione dei vari job che minimizza il makespan per l'esempio descritto, sfruttando l'algoritmo LPT.



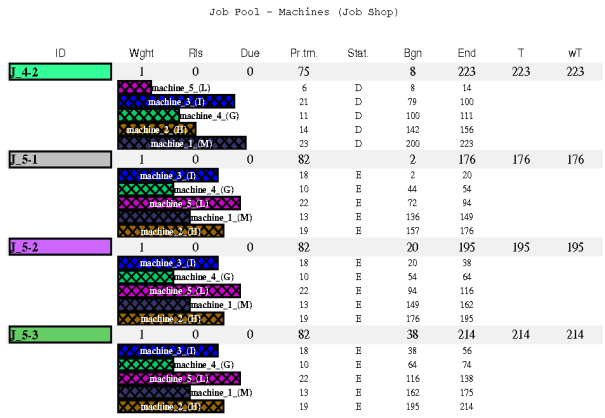


Figura 10: Dettaglio (D) della Figura 7. Il *job pool* del programma mostra il routing tecnologico di ogni job. Sono riportate in dettaglio tutte le caratteristiche dei job, quali tempi di processameto, tempi di rilascio e consegna, pesi, istanti di inizio e fine processo, tardiness e weighted tardiness.

Log Book - Machines *

Schedule	Time	C_{max}	T_{max}	$\sum U_j$	$\sum C_j$	$\sum T_j$	$\sum w_j C_j$	$\sum w_j T_j$
FCFS	1	252	252	14	3143	3143	3143	3143
General SB Routine / C	3	260	260	14	3236	3236	3236	3236
LPT	1	251	251	14	3117	3117	3117	3117
SPT	1	361	361	14	3023	3023	3023	3023

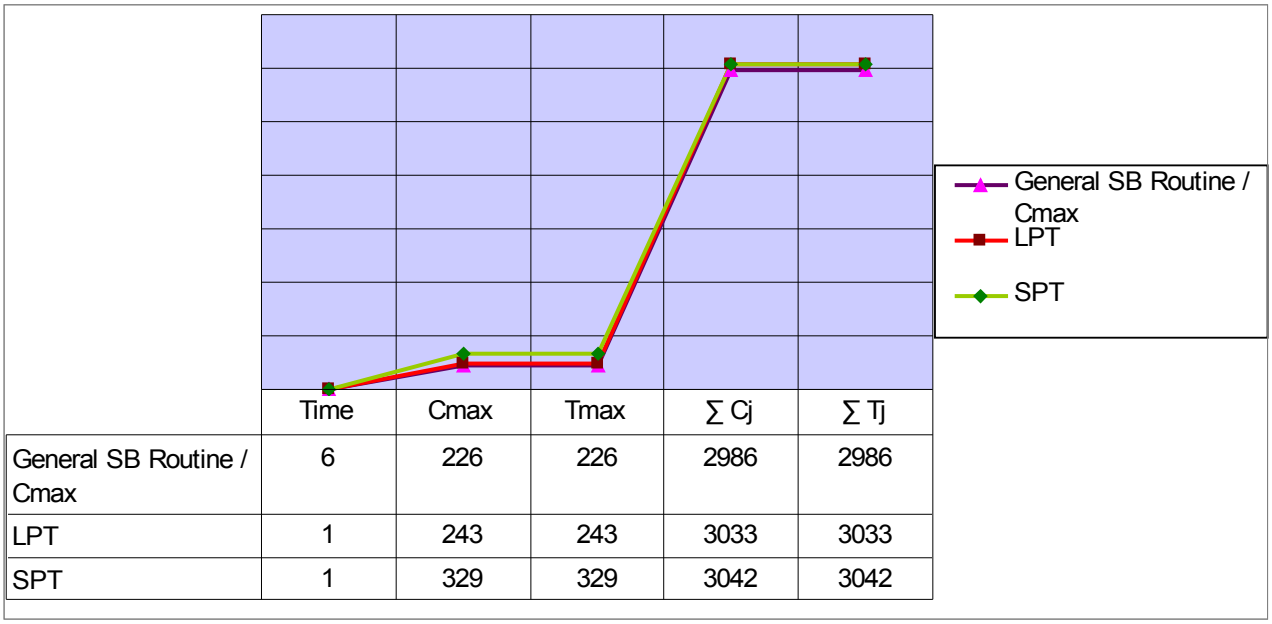
Figura 11: Dettaglio (C) della Figura 7. Il "log book" mostra i risultati ottenuti con i diversi algoritmi testati (per righe). Nelle colonne: tempo di elaborazione impiegato dall'algoritmo, makespan, massimo tardiness, numero dei job in ritardo, total flow time, total tardiness, total weighted flow time e total weighted tardiness.

Dall'analisi dei tempi di processamento in Figura 9, si nota che le isole di produzione con più alti tempi operazione sono la "machine 3 (I)" con processor time pari a 231 unità di tempo e la "machine 1 (M)" con processor time pari a 214 unità di tempo.

Una simile situazione si presta ad un uso del software di pianificazione come simulatore. Cosa accadrebbe con una modifica del layout del sistema produttivo aggiungendo una nuova macchina identica a quella già presente, per ciascuna delle suddette isole I ed M schematizzate in Figura 6.

In questa nuova simulazione, ogni job potrà passare indifferentemente su entrambe le macchine presenti nelle isole "machine 3 (I)" e "machine 1 (M)". Con questa modifica il sistema passa da *Job*

Shop a ***Flexible Job Shop***! Questo consente di ridurre ulteriormente il makespan di processo (Tabella 4) ed aumentare di conseguenza la saturazione del parco macchine ma evidentemente richiede un investimento che deve essere valutato. In particolare, confrontando le Tabelle 3 e 4 si nota che il beneficio risultante è di ridurre da 251 unità di tempo ottenibile con un algoritmo LPT a 226 unità di tempo necessarie per il piano produttivo, ottenuto con l'algoritmo euristico General SB Routine/Cmax.



	Time	Cmax	Tmax	∑ Cj	∑ Tj
General SB Routine / Cmax	6	226	226	2986	2986
LPT	1	243	243	3033	3033
SPT	1	329	329	3042	3042

Tabella 4: Confronto tra i medesimi algoritmi di Tabella 3 e con la stessa commessa ma con la modifica del sistema produttivo a Flexible Flow Shop.

CONCLUSIONI

E' stato proposto un modello largamente impiegato in letteratura scientifica per il problema della pianificazione della produzione.

E' stato proposto un semplice caso applicativo relativo ad una commessa di 5 prodotti.

E' stato descritto il funzionamento di un programma che permette di confrontare i risultati di diversi algoritmi.

Come mostrato nei casi esaminati, il programma permette di ottimizzare una certa funzione obiettivo (nel caso specifico il makespan) simulando diverse situazioni, come ad esempio l'aggiunta di due macchine, con un tempo di elaborazione dell'ordine dei secondi.

Il programma permette di ricavare anche numerose informazioni sul piano di produzione quali, saturazione delle macchine, work in process, numero di job in ritardo, etc.

RINGRAZIAMENTI

Studio sviluppato dagli studenti Fuoco e Puppato nell'ambito dell'insegnamento di Studi di Fabbricazione del Corso di Laurea Specialistica in Ingegneria Meccanica anno accademico 2008-09, presso la Facoltà di Ingegneria dell'Università di Pisa.

Una versione ridotta della relazione è in corso di pubblicazione [18].

BIBLIOGRAFIA

[1] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, J. Weglarz, *Scheduling Computer and Manufacturing Processes*, 2a Ed., Springer, 2001.

[2] G. Kedall, E. Burke, S. Petrovic, M. Gendreou (Eds.), *Multidisciplianry Scheduling: Theory and Applications*, 1st Multidisciplinary International Conference Scheduling Theory and Applications, MISTA '03, Nottingham, UK, 13-15 August 2003. Selected Papers.

[3] Eitan Frachtenberg, Uwe Schwiegelshohn (Eds.), *Job Scheduling Strategies for Parallel Processing,* International Parallel Processing Symposium. 12, 1998, Orlando, Florida, USA.

[4] John F. Proud, *Master Scheduling. A practical guide to competitive manufacturing*, 3rd Ed., John Wiley e Sons, Inc., 2007.

[5] Joseph Y-T Leung, *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, Chapman & Hall/CRC Computer and Information Science Series, 2004.

[6] M. L. Pinedo, *Scheduling: Theory, Algorithms and System*, 3rd Ed., Springer, 2008.

[7] Doc Palmer, *Mantenance Planning and Scheduling HandBook*, 2nd Ed., Mc Graw-Hill, 2005.

[8] M. Zweben, M. S. Fox, *Intelligent Scheduling*, Professional Book Center, 1994.

[9] A. Rossi, G. Dini, *An evolutionary approach to complex job-shop scheduling and flexible manufacturing system scheduling*, J Eng Manuf- Part B 2001; 215: 233-45.

[10] Stecke KE, Raman N, *FMS planning decision, operating flexibilities and system performance*, IEEE Trans Eng Manage, 1995; 42: 82-90.

[11] L. Bianco, M. Caramia, *Metodi quantitativi per il Project Management*, Hoepli, 2006.

[12] F.W. Taylor, *Principles of Scientific Management*, Harper & Brothers, New York, 1911.

[13] F. Xhafa, *Mataheuristics for Scheduling Industrial and Manufacturing Applications*, Springer, 2008.

[14] http://www.mathematik.uni-osnabrueck.de/research/OR/class/, ult. Acc. Gen 2010.

[15] P. Brucker, *Scheduling Algorithms*, 5th Ed., Springer, 2007.

[16] Feldman, Pinedo, *Lekin Sheduler® ver. 2.4*, 2001*, Leonard N. Stern School of Business, New York University,* http://www.stern.nyu.edu/, ult. acc. Dic. 2010.

[17] Jain A.S., Meeran S., *Deterministic Job shop scheduling: past, present and future*, Eur J Op Res, 1999; 113: 390-434.

[18] A. Puppato, B. Fuoco, A. Rossi, M. Lanzetta: *Ottimizzazione della produzione tramite software di scheduling*, Ed. ANIPLA, Italian National Association for Automation, Fiera Milano Editore, ISSN: 0005-1284, Anno LVIII, n. 6, Giugno 2010, pp. 36-40 (5), http://www.ilb2b.it.