# Ottimizzazione della produzione tramite software di scheduling

Alessio Puppato, Barbara Fuoco, Andrea Rossi, Michele Lanzetta

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. Tale approccio consente un miglioramento nella programmazione della produzione attraverso una ottimizzazione delle risorse aziendali. La simulazione della pianificazione è quindi un efficace strumento per mantenere o aumentare la competitività dell'impresa nel medio-lungo termine grazie ad una gestione più efficiente dei processi.

I problemi di pianificazione e scheduling sono problemi decisionali che consistono nella ripartizione nel tempo delle attività da compiere sull'insieme delle risorse disponibili, allo scopo di ottimizzare una o più performance. Tali problemi rivestono un ruolo fondamentale in svariati settori come: produzione, distribuzione e trasporti, informatico, comunicazioni e in tutti quegli ambiti nei quali una migliore gestione delle risorse può portare ad una riduzione dei costi.
I primi problemi di scheduling studiati dai ricercatori negli anni 50 riguardarono l'ottimizzazione della funzione obiettivo *makespan* (tempo di completamento di tutte le attività) e per essa sono stati proposti nel tempo numerosi algoritmi di soluzione sempre più efficienti. Col passare degli anni, i problemi affrontati sono diventati più complessi e i ricercatori non sempre sono stati capaci di sviluppare algoritmi efficienti per la loro trattazione. In seguito, con l'avvento della teoria della complessità computazionale, si è visto che molti di questi problemi erano di per sé di difficile risoluzione e negli anni 70 molti di questi furono classificati come problemi strettamente NP-hard (nondetermistic polynomial-time hard). Tale complessità ha spinto la ricerca verso lo sviluppo di algoritmi euristici [1] che forniscono una soluzione ammissibile quanto più possibile prossima a quella ottimale. Difatti il calcolo della soluzione ottima nei problemi di classe NP-hard può essere computazionalmente troppo onerosa, inoltre possibili approssimazioni nei parametri del modello renderebbero inutile tale sforzo. Dunque nella pratica si accettano soluzioni "buone" e non "ottime".

## Lo scheduling

Lo *scheduling* consiste nel determinare la distribuzione e il sequenziamento ottimale delle attività/operazioni, che devono essere eseguite su una o più risorse, ed ottimizza una certa funzione obiettivo, rispettando al contempo i vincoli imposti. Nel contesto della fabbricazione industriale, con il termine macchina si indica una qualsiasi risorsa come ad esempio: una macchina utensile, un centro di lavoro, un reparto, un operatore ecc. Le attività, che possono consistere in una sola o più operazioni elementari (ciclo di lavorazione), sono indicate con il termine *job*.
Per ottenere la pianificazione ottimale bisogna descrivere i fattori che influenzano un problema di scheduling, ovvero: le caratteristiche e disposizione delle macchine, le caratteristiche dei job e del processo produttivo ed infine il tipo di funzione da ottimizzare.
Le principali configurazioni delle macchine sono [1]: Modello Macchina Singola; Macchine Parallele; Job Shop; Flow Shop (modello di Johnson [2]); Flexible Job Shop; Flexible Flow Shop; Open Shop.
A ciascun job possono essere associate molteplici informazioni [3]:

- *Tempo di processamento (processing time)* $\tau_{ij}$: 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 $\tau_j$.
- *Tempo di rilascio (release date)* $r_j$: 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)* $d_j$: 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

A. Puppato, B. Fuoco, A. Rossi, M. Lanzetta - Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione, Università di Pisa

nome di *deadline*.
- *Peso* $w_j$: 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 esempio costo di immagazzinamento). È 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[4]:

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

*Macchine Parallele:* si possono avere tre tipologie di architettura.
- *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 $\tau_{ij}$ 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.

*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). L'ordine delle operazioni è fisso per ogni job ma i tempi di lavorazione su ogni macchina $\tau_{ij}$ possono essere differenti da job a job.

*Flexible Flow Shop:* è una generalizzazione dei sistemi produttivi di 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.

*Job Shop:* anche in questa architettura vi sono *m* macchine, ma ciascun job ha un proprio ordine con cui visitarle. L'ordine delle operazioni (routing) per ciascun job è fisso ed è diverso da job a job e i tempi di lavorazione $\tau_{ij}$ su ogni macchina possono essere differenti da job a job.

*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.

*Open Shop:* in questo caso 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.

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 $s_{jk}$. 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 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 parametri da associare ai job.

La lettera *i* è utilizzata come pedice per riferirsi alla macchina, mentre con la lettera *j* ci si riferisce al job.
- *Tempo di completamento* $c_{ij}$: indica il tempo (rispetto a un tempo iniziale) necessario affinché 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* $L_j$: 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* $T_j$: 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, quindi 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* $E_j$: 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* $U_j$: è definita come $U_j=1$ se $c_j>d_j$ mentre $U_j=0$ se $c_j \leq d_j$ [5].

## 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.

Deve poi essere definita una *funzione obiettivo*; ecco alcuni esempi: 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*.

Esistono numerosi parametri (e ne vengono introdotti sempre di nuovi) associati alle funzioni obiettivo, di cui i principali sono:

*Makespan o tempo di completamento* $C_{max}$*:* 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* $L_{max}$*:* 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* $\Sigma w_j C_j$*:* 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* $\Sigma w_j T_j$*:* 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* $\Sigma w_j U_j$*:* rappresenta il numero di job in ritardo rispetto alla propria due date ed è spesso utilizzata nella pratica.

*Total setup time* $\Sigma SU_i$*:* 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) [6].

## Algoritmi di simulazione

I principi generali esposti in precedenza (problema di scheduling e ottimizzazione) possono essere estesi ad un particolare caso aziendale.

Nel caso esaminato in questo articolo, l'azienda deve produrre una commessa di 5 prodotti diversi, adottando un *sistema produttivo tipo Job Shop*, dove ogni manufatto ha un determinato routing tecnologico fissato da esigenze meccaniche e di corretto funzionamento dei prodotti stessi (tabelle 1 e 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.

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.

Il layout produttivo rappresentato nella figura 1 è costituito da 5 isole flessibili di produzione e un sistema automatico di movimentazione dei pallet contenenti i prodotti (work in process).

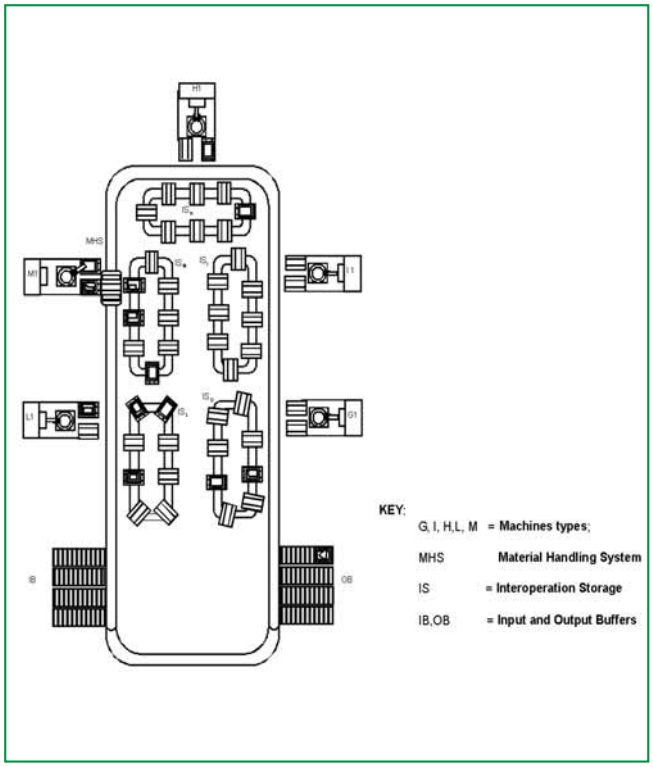



**Figura 1 - Esempio di layout di impianto del tipo Job Shop**

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).

| | Job 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)**

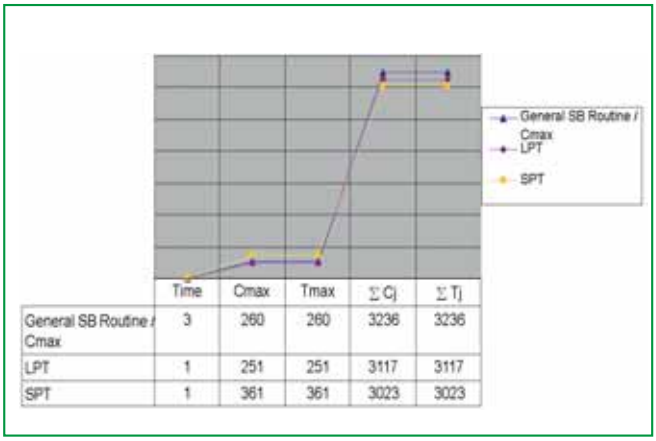
Per la soluzione del problema viene utilizzato il software per la pianificazione automatica dei cicli Lekin Scheduler [7], questo software permette di simulare: i principali sistemi produttivi (elencati sopra), vari algoritmi metaeuristici di ottimizzazione (costruttivi e migliorativi) [1] con la possibilità di aggiungerne di nuovi da parte dall'utilizzatore mediante l'opzione "plug in manager" presente nel programma. I risultati dell'ottimizzazione a singolo o multiobiettivo ottenute con i vari algoritmi sono facilmente leggibili e comparabili grazie agli strumenti grafici del sofware.
Nella figura 2 è illustrata l'interfaccia grafica del programma.


**Figura 2 - Screenshot del software Lekin Scheduler [7] riguardante il caso esaminato, con (A) i dati relativi al prodotto, (B) al sistema produttivo inseriti dall'utente, (C) i risultati ottenuti con i vari algoritmi e (D) il piano calcolato**

Il programma permette di testare diversi algoritmi di ottimizzazione al fine di determinare il minimo della funzione obiettivo, che nel caso esaminato è il makespan.
Anche per quanto riguarda i criteri di ordinamento dei job ne vengono sviluppati sempre di nuovi.
Si riportano i seguenti più noti [8]:

- *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 la macchina che influenza in modo più consistente il valore del makespan rispetto alle altre e tale macchina viene detta "risorsa 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.
  Il ruolo di risorsa bottleneck è rivestito da una macchina diversa ad ogni passo, pertanto l'algoritmo è detto shifting bottleneck.

Gli algoritmi citati sono stati testati sul caso in esame tramite il programma descritto e i risultati ottenuti utilizzando i diversi criteri di ordinamento sono sintetizzati nella figura 3.



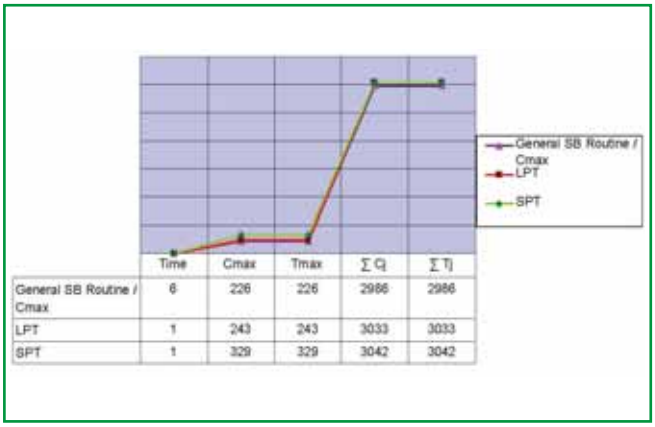

| | Time | Cmax | Tmax | ΣCj | ΣTj |
|---|---|---|---|---|---|
| General SB Routine / Cmax | 3 | 260 | 260 | 3236 | 3236 |
| LPT | 1 | 251 | 251 | 3117 | 3117 |
| SPT | 1 | 361 | 361 | 3023 | 3023 |

**Figura 3 - Confronto tra i tre algoritmi di ottimizzazione testati (dall'alto): General shifting bottleneck, Longest processor time, Shortest processor time.**
**Nelle colonne (da sinistra): tempo di calcolo in secondi, makespan, tardiness, il total makespan ed il total tardiness**

Dall'analisi dei tempi di processamento (zona B nella figura 2) si nota come le isole di produzione con più alti tempi operazione siano la "machine 3 (I)" con processor

**Figura 4 - Il "job pool" del programma mostra il routing tecnologico di ogni job. Sono riportati in dettaglio tutte le caratteristiche dei job, quali, tempi di processamento, tempi di rilascio e consegna, pesi, istanti di inizio e fine processo, tardiness e weighted tardiness (A)**

time pari a 231 unità di tempo e la "machine 1 (M)" con processor time pari a 214 unità di tempo. Si propone quindi una modifica del layout del sistema produttivo aggiungendo una nuova macchina identica a quella già presente per ciascuna delle suddette isole. In questa nuova simulazione, ogni job potrà quindi passare indifferentemente su entrambe le macchine presenti nelle isole "machine 3 (I)" ed "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 (figura 5) ed aumentare di conseguenza la saturazione del parco macchine ma evidentemente richiede un investimento che deve essere valutato.
In particolare, confrontando le figure 4 e 5 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 |

**Figura 5 - I medesimi algoritmi della figura 3, con la stessa commessa ma con la modifica del sistema produttivo a Flexible Flow Shop**

## Conclusioni

Il caso applicativo trattato consente di evidenziare i vantaggi derivanti dall'uso di un software di scheduling anche per i neofiti. Esistono numerosi programmi in commercio ma è anche possibile svilupparli autonomamente in base alle proprie esigenze, selezionando tra le opzioni brevemente descritte in questo articolo. Con tali programmi è possibile ricavare oltre al makespan qui trattato, numerose altre informazioni quali il numero di job in ritardo, il work in process, la saturazione delle machine e le altre descritte precedentemente. Ciò consente una produzione più efficiente per quel che riguarda le quantità da produrre ed il rispetto dei tempi di consegna, con conseguente incremento del rendimento aziendale e della soddisfazione dei clienti.
I tempi per l'elaborazione di ciascuna configurazione produttiva su PC sono dell'ordine dei secondi (figura 3 e figura 5), quindi i software di scheduling sono anche uno strumento efficace per valutare la necessità e la convenienza di dotarsi di nuove macchine ed apparecchiature.
Le aziende che utilizzano queste tecniche sono più reattive e dinamiche e riescono ad affrontare più prontamente le congiunture di un mercato sempre più esigente e a domanda variabile.

## Bibliografia

[1] L. Puppato, B. Fuoco, A. Rossi, M. Lanzetta, *Scheduling: modelli, algoritmi e simulazione*, Atti del Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione, Università di Pisa, DIMNP(2010)001, Gennaio 2010, 25 pagg. http://eprints.adm.unipi.it/716.
[2] G. Kedall, E. Burke, S. Petrovic, M. Gendreou (Eds.), "Multidisciplianry Scheduling: Theory and Applications", *Mista '03* Nottingham, Selected Papers, UK, 13-15 August 2003.
[3] D. Palmer, *Maintenance Planning and Scheduling HandBook*, 2nd Ed., Mc Graw-Hill, 2005.
[4] M. Zweben, M. S. Fox, *Intelligent Scheduling*, Professional Book Center, 1994.
[5] Stecke K.E., Raman N., "FMS planning decision, operating flexibilities and system performance", *IEEE Trans Eng Manag*, 1995; 42:82-90.
[6] F.W. Taylor, *Principles of Scientific Management*, Harper & Brothers, New York, 1911.
[7] Feldman, M.L. 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.
[8] Jain A.S., Meeran S., "Deterministic Job shop scheduling: past, present and future", *Eur J Op Res*, 1999; 113: 390-434.

La versione estesa dell'articolo è disponibile in Internet [1]. Il presente studio è stato 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. Il Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione sviluppa queste ed altre tematiche anche con aziende, tra cui la Società Piaggio di Pontedera (PI).