L'ottimizzazione dei programmi per le linee di taglio longitudinali (slitter), spianatura e cesoiatura, dovuta alle esigenze e richieste dei Centri Servizi dell'Acciaio e delle divisioni "taglio coil e lamiera" di diverse acciaierie: ecco un nuovo approccio risolutivo

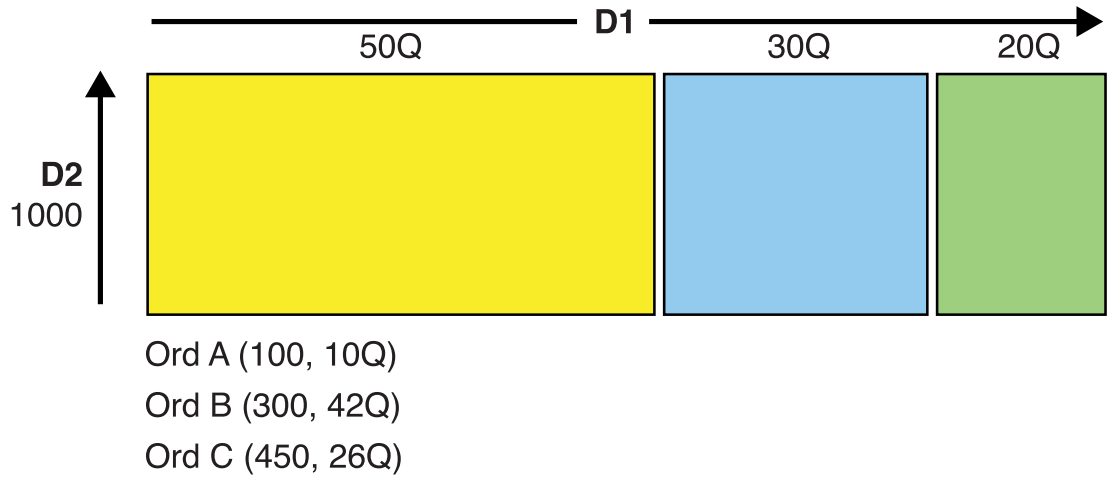
Emilio Bertolotti, Gino Giannone

Ottimizzare la programmazione di linee taglio e spianatura

UNA PROBLEMATICA RICORRENTE

È ormai da anni che incontrando il responsabile di un centro servizi o della programmazione coil di una acciaieria per discutere delle loro problematiche di produzione, capiti di assistere all'esternazione di un cruccio piuttosto ricorrente che suona più o meno così: *«Il mercato è diventato difficile, i clienti richiedono un livello di servizio sempre più gravoso da soddisfare e per andargli addietro rischiamo ogni giorno di intaccare ulteriormente la marginalità: date di consegna stringenti, tolleranze su quantità ordinate da rispettare rigidamente, vincoli dimensionali sui colli... Il nostro listino poi, è*

FIG|01| Esempio introduttivo.



molto condizionato dalla concorrenza ed è difficile recuperare margine attraverso la leva d i prezzi. Abbiamo ormai razionalizzato il ciclo produttivo un po' ovunque e risulta difficile pensare ad altri interventi significativi nel breve. In effetti, una delle poche aree in cui non siamo ancora riusciti ad ottimizzare realmente i processi e dove sono convinto esista un margine di miglioramento è l'area della programmazione degli ordini e della produzione. Alla programmazione lavorano persone con parecchi anni di esperienza e che conoscono molto bene il loro lavoro. Del resto abbiamo sperimentato sul campo quanto sia preziosa la loro esperienza e quanto sia ragguardevole il tempo necessario per formare un buon programmatore partendo da zero. Tuttavia,la gestione della programmazione sta diventando sempre più difficile anche per loro: il mercato diventa sempre più dinamico, le richieste dei clienti più articolate, il servizio al cliente sempre più importante e la possibilità di sfruttare al meglio le macchine, il materiale disponibile, i processi è una questione fondamentale di sopravvivenza. Senza contare i rischi che corriamo quando uno di questi signori è in ferie e restiamo con un organico pericolosamente sottodimensionato».

A questo punto del dialogo, di solito chiediamo: *«Avete mai pensato di potenziare gli strumenti che utilizzano gli esperti della programmazione per agevolarli nel gestire e controllare un processo così importante che potrebbe innegabilmente consentire un recupero di efficienza significativo per l'intera azienda?»*.

La risposta che si ottiene è immancabilmente la seguente: *«Si, ci abbiamo pensato. Qualche anno fa abbiamo deciso di provare a supportare la pianificazione con strumenti informatici. Abbiamo dapprima provato ad applicare uno di quei pacchetti di "schedulazione della produzione" che esistono sul mercato. In teoria sembrava sufficientemente flessibile, ma in pratica non si è dimostrato adattabile alle esigenze specifiche della programmazione delle linee di taglio. Di conseguenza abbiamo cercato di costruire qualche cosa internamente che migliorasse comunque il supporto informativo per i programmatori. Il risultato è che oggi abbiamo sicuramente migliorato la situazione con questi nuovi strumenti, ma il grosso della programmazione resta comunque sulle spalle dei nostri programmatori con tutte le conseguenze e le criticità del caso»*.

Questo tipo di dialogo esemplifica una problematica ben nota agli addetti ai lavori: la convinzione che ci siano margini di miglioramento in un processo aziendale così strategico come la programmazione, ma che nella pratica non sia altrettanto semplice ottenere questi miglioramenti. Perché è così difficile ottimizzare in modo sistematico la programmazione delle linee di taglio e spianatura? Quali sono le specificità e le complessità che si celano dietro questo apparentemente "normale" problema di programmazione? È possibile pensare a un metodo, un modello, uno strumento che consenta di ottimizzare sistematicamente la programmazione e pianificazione garantendo che le scelte fatte siano sempre le migliori possibili? In ultima analisi, è possibile una ottimizzazione della programmazione e pianificazione che consenta un miglior controllo dell'intero processo, un migliore servizio ai clienti e un recupero di marginalità con una sistematica razionalizzazione nell'utilizzo delle risorse coil e macchine?

SULLE TRACCE DELLA COMPLESSITÀ

Procediamo con ordine e iniziamo a capire in modo puntuale perché il problema sia così complesso, perché sia umanamente impossibile gestire tutta questa complessità e quali siano le conseguenze pratiche di queste limiti. Infine analizzeremo se e come la matematica e la tecnologia attualmente disponibili possono concretamente aiutare a migliorare lo

stato attuale delle cose.

Partiamo da un esempio introduttivo di programmazione, molto semplice, che tuttavia già contiene gli ingredienti necessari a familiarizzare con la complessità di tipo "combinatorio" che il problema della pianificazione delle linee di taglio nasconde.

Nell'esempio di **fig. 1** sono raffigurati tre ordini da programmare (ord A, ord B, ord C) e tre rettangoli che rappresentano tre coil disponibili e qualitativamente compatibili (stesso prodotto, stesso spessore, stessa qualità o qualità compatibile) con gli stessi ordini A, B, C. La dimensione D1 rappresenta di fatto la "lunghezza" del coil anche se di solito è preferibile utilizzare il peso come unità di misura per questa dimensione (il fatto che le larghezze dei coil siano tra loro identiche rende possibile utilizzare semplicemente il peso come unità di misura, diversamente avremmo dovuto utilizzare il peso per unità di larghezza). Secondo i valori indicati in **fig. 1** il primo coil pesa 50 q, il secondo 30 q e il terzo 20 q. La dimensione D2 rappresenta invece la "larghezza" o ampiezza del coil.

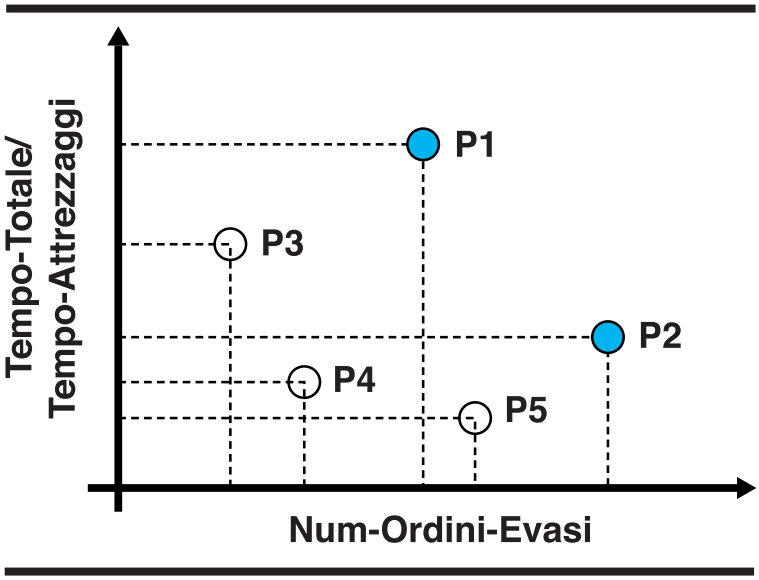
Questo semplice problema di programmazione, costruito *ad hoc* per scopi espositivi, consiste quindi nel trovare il "miglior programma" di produzione che soddisfi i tre ordini Ord A, Ord B, Ord C con le loro larghezze richieste (largh-A = 100 mm, largh-B = 300 mm, largh-C = 450 mm) e pesi richiesti (peso-A = 10 q, peso-B = 42 q, peso-C = 26 q) considerando i coil a disposizione. Trascuriamo, solo per il momento, le numerose "variabili addizionali" che dovrebbero essere considerate in un reale esempio di programmazione quali: ulteriori vincoli dimensionali sugli ordini (ad es. massimo peso collo), vincoli delle macchine (ad es. numero massimo tagli), vincoli sulla qualità alternative del materiale, tolleranze di peso, vincoli sulle date di consegna, eventuali necessità di integrazione con la fase di spianatura, difettosità sui coil e così via. Riprenderemo più avanti il discorso sul rispetto di tutti questi vincoli, che rendono sicuramente molto più difficile la programmazione dell'esperto umano, ma che ai fini del calcolo delle combinazioni possibili, non hanno una incidenza determinante, se non quella di "ridurre" il numero di possibilità. Vedremo inoltre come questa "riduzione" non sia purtroppo così significativa rispetto al numero totale di combinazioni che comunque restano da analizzare.

Abbiamo virgolettato il termine "miglior programma" in quanto questa definizione merita diversi chiarimenti. Esistono infatti molteplici criteri, che chiameremo criteri di ottimizzazione e che definiscono l'efficacia e l'efficienza di un programma. Una tipica lista dei principali criteri di ottimizzazione di un programma è la seguente:

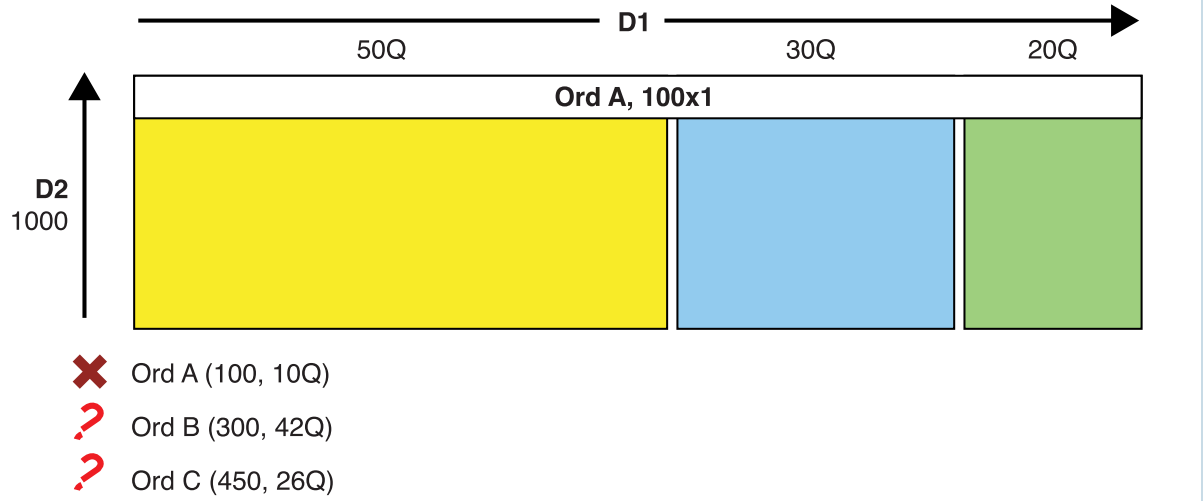
– rispetto delle date di consegna degli ordini inseriti nel programma e in portafoglio;
– minimizzazione dello sfrido;
– minimizzazione del numero di attrezzaggi;
– rispetto delle tolleranze sulle quantità ordinate;
– evitare di lavorare coil solo parzialmente;
– minimizzazione dei tagli trasversali (battute);
– cercare di completare gli ordini inseriti in programma (evitare residui, a meno che non siano ordini aperti o ripetitivi);
– integrare/sincronizzare i piani di spianatura;
– scelta della macchina migliore sia tenendo conto delle ottimizzazioni contenute nei criteri precedenti sia cercando di bilanciare il carico delle diverse macchine (schedulazione). Va notato che tra i criteri di ottimizzazione, non sono inclusi i vincoli tipo diametro minimo o caratteristiche di macchina che abbiamo invece considerato tra le cosiddette "variabili addizionale", ossia tra i vincoli da rispettare comunque e sempre in modo da garantire che un programma sia un programma fattibile, ossia producibile.

È una lista piuttosto lunga, senza contare che alcuni di questi criteri sono addirittura in contrasto tra di loro, quindi ciò che consideriamo il "miglior programma" risulta di fatto necessariamente dipendere dal particolare contesto in cui il programma viene formulato. Ad esempio, se si è sotto "stress di consegna" con un notevole numero di ordini da evadere, converrebbe magari privilegiare programmi che completino molti ordini nel breve tempo rispetto a quelli "lunghi o di maggior peso" che ottimizzino gli attrezzaggi. In pratica ciò che consideriamo il "miglior programma" può essere interpretato come una scelta opportunistica tra una serie di "buoni" programmi candidati. Il problema vero è quello di considerare tutti i cosiddetti "buoni" programmi candidati senza ignorare

FIG|02| Programmi buoni dominanti.



FIG|03| Primo tentativo.



delle opportunità. Perdersi opportunità significa infatti il più delle volte penalizzare efficacia ed efficienza dell'intera produzione e in ultima analisi impattare servizio e margini.

È possibile arrivare ad una definizione precisa di "buon" programma: un "buon" programma è un programma che per almeno uno dei criteri di ottimizzazione predefiniti supera tutti gli altri programmi fattibili. Il cuore del significato è in effetti contenuto nella frase "tutti i programmi fattibili" in quanto, come vedremo, proprio questo fattore costituisce la principale fonte di complessità che rende estremamente difficile garantire una vera programmazione ottimizzata. Per proseguire nel nostro ragionamento e per esemplificare tale definizione, supponiamo, solo per il momento, di essere in grado di esplorare tutti i programmi fattibili e di considerare inoltre solo due criteri di ottimizzazione: attrezzaggi (ossia tempo totale macchina diviso il tempo di attrezzaggio) e numero di ordini evasi. In tale situazione, se l'insieme dei programmi fattibili si riduce a due soli programmi, P1 e P2, dove P1 sia migliore dal punto di vista dei tempi di attrezzaggio e peggiore dal punto di vista del numero di ordini evasi rispetto a P2 allora entrambi P1 e P2 sarebbero da considerare "buoni" programmi secondo la definizione. Se si aggiungesse un terzo programma fattibile P3 peggiore sia dal punto di vista dei tempi di attrezzaggio sia per ordini evasi rispetto a P1 allora non si deve considerare P3 un "buon" programma in quanto "surclassato" da P1. La definizione di "buon" programma equivale in pratica alla relazione di "dominanza" definita su un numero di dimensioni pari al numero di criteri di ottimizzazione predefiniti. Considerando i due criteri utilizzati nell'esempio, possiamo illustrare la relazione con il grafico bidimensionale di **fig. 2** con 5 programmi fattibili.

Nel grafico di **fig. 2** sia P3 che P4 sono "surclassati" da P1 (P3 non è surclassato da P2), ossia considerando esclusivamente i nostri due criteri "tempi attrezzaggio" e "numero ordini evasi", non avrebbe nessun senso preferire P3 o P4 a P1. Parimenti, P4 e P5 sono "surclassati" da P2 (P5 non è surclassato da P1). P1 e P2 non sono "surclassati" da nessun altro programma e quindi entrambi sono da considerare "buoni" programmi (stiamo sempre assumendo che P1, P2, P3, P4, P5 siano tutti i programmi fattibili). Per concludere quindi torniamo alla domanda originale: quale e' in questo esempio il "miglior" programma? La risposta è: in un periodo di "stress di consegna" è probabilmente preferibile P2, diversamente potrei preferire P1. Inoltre, siccome stiamo considerando tutti i programmi fattibili siamo sicuri che questa sia la scelta ottima (siamo certi che non sia possibile fare di meglio).

Dopo questo chiarimento sulla definizione di "miglior programma" torniamo al nostro esempio introduttivo e cominciamo a costruire delle soluzioni possibili che giudicheremo buone o meno sia sulla base dei criteri sopraelencati sia tenuto conto della relazione di dominanza. Seguiremo un metodo piuttosto "naive" nel generare i nostri tentativi perchè questo approccio ci aiuterà a dare un'idea di quante siano potenzialmente le strade o le combinazioni da considerare e quanto sia difficile essere sicuri di analizzare tutte le possibilità. Il primo tentativo (**fig. 3**) consiste nel programmare un nastro da 100 mm sui tre coil in sequenza e completare così l'ordine A cercando contemporaneamente, se possibile, di allungare il più possibile l'utilizzo dello stesso schema di taglio (riduzione tempi di attrezzaggio). La somma dei pesi dei tre singoli nastri da 100 mm ottenuti dai tre coil ammonta esattamente a 10 q (50 q + 30 q + 20 q * 100 / 1000 = 10 q). Di nuovo, trascuriamo per il momento le altre "variabili addizionali" quali ad esempio il controllo che i pesi dei diversi coil consentano di rispettare almeno i vincoli di peso minimo dei nastri risultanti. Purtroppo,

questo primo tentativo, inizialmente attraente, non conduce a nulla di buono. Infatti per combinare le ampiezze e i pesi dei restanti ordini B e C si è costretti ad inserire inopportuni e costosi tagli trasversali (battute) o a introdurre uno sfrido inammissibile oltre che a generare residui coil e ordini.

Nel secondo tentativo (**fig. 4**) proviamo di nuovo con l'ordine A aumentando il numero di strisce e programmiamo 2 nastri da 100 mm solo sul primo coil. In tal modo otteniamo facilmente, di nuovo, il completamento dell'ordine A. La semplice strategia (euristica) che stiamo seguendo attraverso i vari tentativi cerca di partire con un ordine il cui rapporto peso/larghezza è tra i più bassi inserendo il minor numero di strisce possibili. In tal modo si privilegia un obiettivo ben preciso: cercare implicitamente di aumentare il più possibile il peso dei programmi ottenuti per minimizzare i tempi di attrezzaggio. In realtà è una delle tante strategie *(euristiche)* utilizzabili, che come vedremo in seguito, hanno tutte il difetto di non garantire affatto una generazione sistematica delle migliori soluzioni possibili. Anche in questo secondo tentativo non facciamo molta strada, le combinazioni risultanti nel cercare di abbinare gli altri due ordini restanti B e C sono tutt'altro che buone per le stesse ragioni del precedente tentativo (costringono a tagli trasversali, utilizzi parziali...).

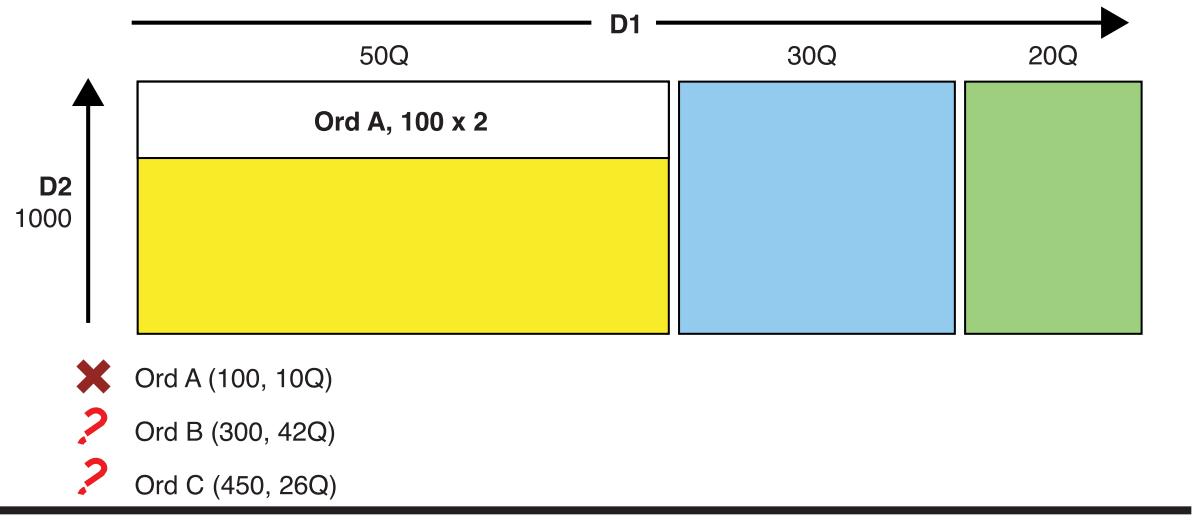
Terzo tentativo (**fig. 5**). Questa volta riproviamo sempre con l'ordine A, una striscia, ma invertiamo anche la sequenza tra secondo coil e terzo coil, perché sembra in tal modo possibile eliminare i tagli trasversali (battute) che erano necessari per completare il programma in entrambi i precedenti tentativi.

Con il cambio di sequenza dei coil il risultato ha successo e otteniamo un programma completo decisamente buono sia dal punto di vista dello sfrido (stiamo trascurando per semplicità lo sfridio fisiologico di 3 / 10 mm che andrebbe comunque considerato nei casi reali) sia dal punto di vista degli schemi di taglio utilizzati, senza tagli trasversali e senza residui coil o ordini (**fig. 6**).

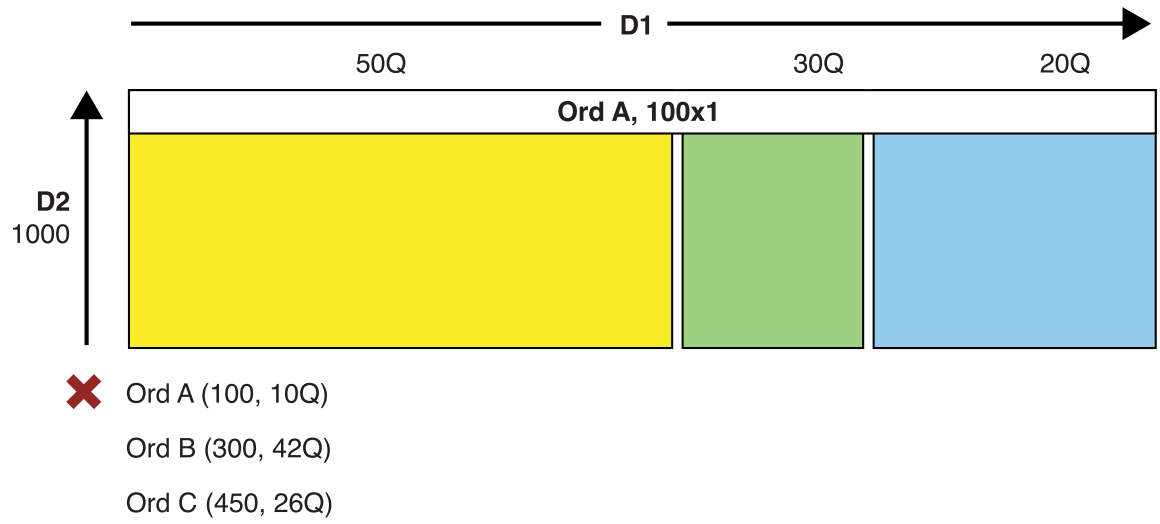
L'esempio è stato costruito *ad hoc* e quindi il risultato porta ad una coincidenza perfetta tra pesi ordini e pesi coil, ma nella reale attività di programmazione molto spesso il problema diventa non tanto quello di centrare perfettamente il peso, ma piuttosto di rispettare le tolleranze ammesse da quel cliente o comunque di avvicinarsi il più possibile al peso richiesto. Questo non cambia tuttavia minimamente le considerazioni sul numero dei tentativi necessari (semmai aumenta le combinazioni possibili).

Questo modo di procedere, per tentativi, sta di fatto alla base di tutte le diverse strategie (euristiche) normalmente utilizzate nelle sessioni di programmazione. Un altro metodo più spesso impiegato nella pratica consiste, ad esempio, nel partire direttamente da abbinamenti di larghezze ordini/strisce che consentano un basso sfrido per poi passare ad abbinare i coil in funzione delle quantità richieste (pesi). Vista la loro diffusione, sarebbe a questo punto importante capire quanto buone siano queste strategie e se ve ne siano alcune migliori di altre. In particolare, sarebbe utile capire se esistono strategie in grado di produrre in modo sistematico i programmi migliori. Più precisamente, vorremmo una risposta alla seguente domanda: "esiste una strategia costruttiva che consenta sistematicamente di generare tutti i programmi "buoni", senza perdersi opportunità e senza essere costretti a considerare tutte le possibili combinazioni e tutti i programmi fattibili?" (ricordiamo che, come visto in precedenza il concetto di "buon" programma è definibile in modo rigoroso a partire da tutti i programmi fattibili una volta definiti formalmente i criteri di ottimizzazione).

FIG|04| Secondo tentativo.



FIG|05| Terzo tentativo.



La risposta a tale domanda è che purtroppo tale strategia o metodo non esiste (questo risultato deriva dalla matematica discreta e dalla teoria della complessità che hanno studiato in modo rigoroso questo tipo di problemi di natura combinatoria). Qualsiasi strategia (euristica), inclusa quella usata da noi nell'esempio, non riesce a coprire tutte le possibilità interessanti ed è anzi provato che alcune strategie privilegiano taluni obiettivi o criteri di ottimizzazione, ma sacrificano sistematicamente altri obiettivi o criteri di ottimizzazione (in sostanza, tali strategie perdono opportunità, che tradotto in termini economici vuol dire perdere margine e servizio).

Visto che non esiste una strategia o un metodo certo in grado di generale le migliori combinazioni, non resta che chiederci se sia possibile capire quante siano queste combinazioni e se sia ragionevole pensare di poterle esplorare a tappeto per essere sicuri di scegliere la strada migliore. La domanda immediatamente successiva è quindi: "quale è il numero delle possibili combinazioni in casi reali in cui siano presenti un certo numero di ordini (quantità che indichiamo con m) e un certo numero di coil (quantità che indichiamo con n)?". Inoltre, supponendo di calcolare questo numero, "è possibile costruire un metodo e un rispettivo programma in grado di generare automaticamente e in tempi ragionevoli tutte queste combinazioni selezionando quelle "buone" per poter poi far scegliere e validare opportunisticamente la migliore soluzione al programmatore?" Se la risposta a quest'ultima domanda fosse "Si", allora avremmo risolto il problema, non si perderebbero opportunità e saremmo sicuri di programmare al meglio, garantendo il massimo del servizio possibile senza sacrificare la marginalità.

Cerchiamo di rispondere alla prima parte della domanda sul numero di combinazioni teoriche per poi illustrare come una tecnica sviluppata specificatamente per questo tipo di problema negli ultimi 10 anni consenta, considerata la potenza di calcolo disponibile nei normali computer odierni (PC o server), di rispondere affermativamente alla seconda domanda per situazioni industriali con un numero di coil e numero di ordini che un centro servizio o una acciaieria si debba trovare a programmare normalmente in una data sessione di lavoro.

Cominciamo a considerare gli ordini e a cercare di capire quante combinazioni che includano almeno uno di questi "m" ordini sia possibile generare trascurando, per il momento, le possibili combinazioni di coil da utilizzare. Partiamo dall'assunto che ogni generica combinazione contente un certo numero di ordini ha in realtà due possibilità: o contiene un dato ordine i-esimo O_i o non lo contiene. Sulla base di questa premessa ed estendendola a tutti i nostri m ordini, il numero totale di combinazioni possibili coincide praticamente con tutti i possibili sottoinsiemi che si riescono a generare a partire da un insieme di m elementi distinti. Ad esempio se ho tre ordini ord-A, ord-B, ord-C, allora m = 3 e i possibili sottoinsiemi di ordini, ossia le possibili combinazioni sono:

– combinazione C1 che contiene solo: ord-A
– combinazione C2 che contiene solo: ord-B
– combinazione C3 che contiene solo: ord-C
– combinazione C4 che contiene solo: ord-A, ord-B
– combinazione C5 che contiene solo: ord-A, ord-C
– combinazione C6 che contiene solo: ord-B, ord-C
– combinazione C7 che contiene solo: ord-A, ord-B, ord-C.

Generalizzando il calcolo, dalla matematica discreta sappiamo che il numero di sottoinsiemi (non vuoti) generabili a partire da un insieme di m elementi distinti è di $2^m - 1$ ("power set" di m). Nel precedente esempio di tre ordini: $2^3 - 1 = 8 - 1 = 7$. Questo numero totale di combinazioni di ordini che

indicheremo con *M* vale quindi: $M = 2^m - 1$.

D'ora in poi tralasceremo nei nostri conti il –1 che complica le notazioni e influenza in modo del tutto marginale il numero totale di combinazioni che ci interessa calcolare. È invece rilevante tener conto di un ulteriore fattore che incide sul numero di combinazioni: la molteplicità delle strisce per un dato ordine (consideriamo solo strisce e non fogli in quanto, dal punto di vista combinatorio l'aspetto fogli è sempre riconducibile alla stessa problematica del taglio strisce/nastri). Infatti va tenuto conto del fatto che per ogni ordine Oi si possono generare combinazioni diverse e quindi programmi diversi in quanto contenenti un numero di strisce diverse dello stesso ordine: ad esempio un programma che contiene l'ordine A con due strisce è sostanzialmente diverso da un programma che contiene lo stesso ordine A con tre strisce. Diciamo che il numero possibile di strisce per ogni ordine Oi può variare da 1 a k (Dove k è un numero intero che esprime il rapporto tra la larghezza del coil e la larghezza dell'ordine. Ad esempio per un coil di larghezza 1000 e un ordine di larghezza 300 il numero di strisce può essere solo 1, 2 o 3). Se definiamo quindi un valore medio di *k*, che indicheremo con *K* maiuscolo, il numero di medio di elementi da considerare non sarà più semplicemente *m*, ma il prodotto *K m*. Quindi il numero totale di combinazioni a partire da *k m* elementi diventa: $M = 2^{Km}$.

Avendo calcolato le possibili combinazioni di ordini e strisce, dobbiamo ora capire come queste si combinano con le possibili combinazioni di coil per arrivare al numero totale teorico di programmi possibili. Procediamo quindi calcolando dapprima il numero di combinazioni di coil possibili per poi capire come questo numero si debba correlare con le possibili combinazioni di ordini che abbiamo appena calcolato. Va notato che a parità di coil utilizzati anche la loro sequenza influenza in realtà la qualità dei programmi ottenuti: due sequenze diverse degli stessi coil potrebbero dar luogo a due programmi diversi se i coil hanno diverso peso, come visto nell'esempio della sessione precedente. Questo aspetto viene per il momento trascurato e discusso in seguito perchè , come vedremo, non costituisce un ostacolo pratico ai fini della reale complessità combinatoria del nostro problema specifico.

Analogamente a quanto fatto per gli ordini, calcoliamo il numero totale di combinazioni di *n* coil, che indicheremo con *N*. Anche per i coil valgono le considerazioni già fatte per gli ordini ed ogni generica combinazione può: o contenere o non contenere un dato coil. Ripercorrendo quindi il ragionamento fatto in precedenza, il numero totale delle possibili combinazioni è uguale al numero di sottoinsiemi *(non vuoto)* generabili a partire da un insieme di *n* elementi distinti, ossia: $N = 2^n$.

Ora che conosciamo il numero di combinazioni di ordini e il numero di combinazioni di coil dobbiamo in qualche modo collegare i due insiemi e trovare il numero totale di combinazioni ossia il numero totale di programmi teorici. Come? Basta fare una semplice considerazione: non esiste nessuna correlazione che vincoli la scelte tra le combinazioni di ordini e la scelta tra la combinazione di coil, questo significa che una combinazione di ordini può in teoria essere abbinata a qualsiasi combinazione di coil, salvo poi verificare la fattibilità dal punto di vista di quelle che abbiamo definito come "variabili addizionali". Quindi il numero totale di combinazioni (che a questo punto potremmo già chiamare programmi teorici) è il prodotto tra il numero totale di combinazioni ordini *M* ed il numero totale di combinazioni coil *N*. Se indichiamo quindi con P il numero totale dei programmi teorici possibili, vale la seguente relazione: $P = M\,N = 2^{km}\,2^{n} = 2^{km+n}$.

Questa quantità è un numero enorme. Considerando a titolo esemplificativo 50 ordini e 50 coil (insieme di ordini e coil, compatibili in termini di prodotto qualità, da programmare in una data sessione di lavoro) e assumendo un *K* medio uguale a 5, otteniamo: $P = 2^{5*50+50} = 2^{300}$.

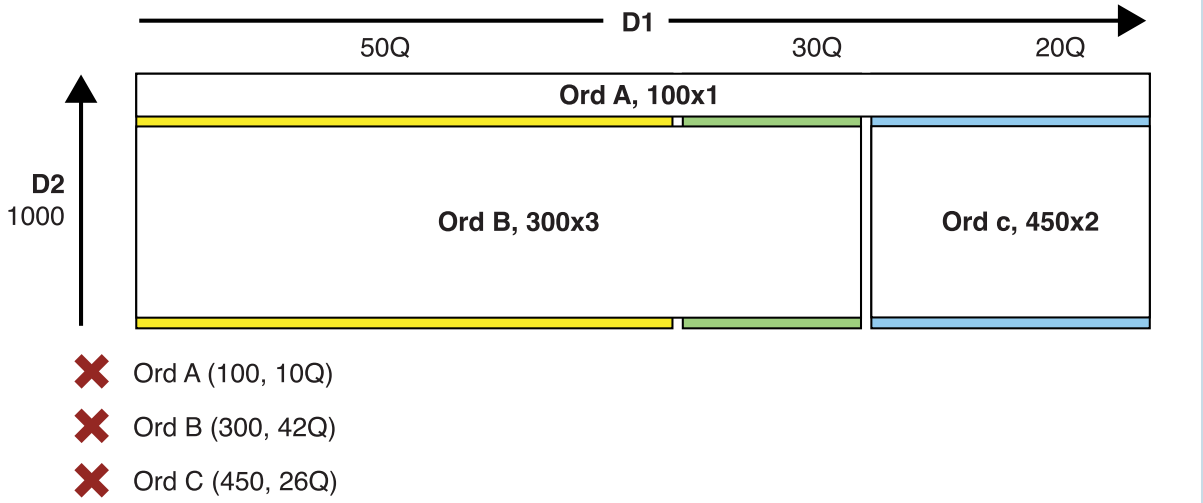
Certo, questi sono tutti i programmi "teorici" e molti di questi sono da scartare se si considerano i vincoli imposti dalle "variabili addizionali" di peso, vincoli nastri, macchine …. Ma, come vedremo nella prossima sessione, anche con una drastica riduzione del numero di programmi teorici, il numero di programmi realmente possibili resta purtroppo una quantità incredibilmente alta. Inoltre, la cosa peggiore è che se si aggiunge qualche ordine e/o qualche coil, come si vede dalla equazione di *P*, il numero totale di combinazioni tende ad aumentare in modo "esponenziale". In pratica all'aumentare di qualche unità delle variabili indipendenti (ordini e coil) le combinazioni aumentano di milioni di miliardi!

LE TECNICHE RISOLUTIVE TRADIZIONALI

Ora sappiamo con che numero di combinazioni abbiamo a che fare ed è giunto il momento di riprendere alcuni dei dettagli accantonati in precedenza, trarre alcune considerazioni sulle tecniche tradizionali, iniziare a rispondere alle nostre domande iniziali sul possibile aiuto da parte della matematica e della tecnologia nel trovare un nuovo ed efficace metodo risolutivo.

1) La prima considerazione riguarda la questione di quelle che abbiamo definito "variabili addizionali" (ad esempio i vincoli su pesi nastri, tolleranze quantità...) che abbiamo sinora accantonato. Tener conto di questi ulteriori vincoli ci consente effettivamente di ridurre di parecchio il numero di combinazioni teoriche. Infatti se pensiamo ad esempio a un vincolo tipo il minimo peso nastro definito per un certo ordine, questi

FIG|06| Terzo tentativo completato.



consente di scartare magari parecchie tra le combinazioni teoriche di coil e anche molti degli abbinamenti con ordini che abbiano valori di peso nastro minimo e massimo molto diversi. La stessa cosa vale per le altre "variabili addizionali" quali ad esempio il massimo numero di tagli effettuabili su una macchina. Tuttavia, la natura esponenziale della equazione di *P* che individua il numero totale di programmi teorici, è tale da vanificare questi pur utili tipi di "riduzione". Infatti se, nell'esempio 50 ordini 50 coil, ipotizziamo di dividere per un miliardo di volte le combinazioni grazie alla "riduzione" generata dalle "variabili addizionali" (non sottrarre un miliardo di combinazioni, bensì dividere le combinazioni teoriche per un miliardo) otterremmo: $P = 1 / 1.000.000 * 2^{300} = 1 / 2^{30} * 2^{300} = 2^{-30} * 2^{300} = 2^{270}$, che non cambia nella sostanza l'entità del numero (basti pensare che già 2^{100} è un numero con 31 zeri!). Quindi la speranza di rendere il problema trattabile grazie alla "riduzione" possibile con l'introduzione delle "variabili addizionali" non è una soluzione percorribile (ovviamente è comunque necessario e fondamentale tenere conto di questi tipi di "variabili addizionali" in sede di programmazione). L'altro aspetto finora trascurato e relativo ai diversi possibili ordinamenti (sequenze) di una data combinazione di coil, ha naturalmente anch'esso impatti sia sul calcolo del numero teorico di combinazioni sia sulla programmazione pratica. Tuttavia, dal punto di vista teorico, considerare i diversi ordinamenti di una data combinazione di coil non fa che aumentare ulteriormente il numero di combinazioni possibili e quindi rendere il problema più complesso. Tuttavia, nella pratica la scelta dell'ordinamento ha effetto solo su una parte degli obiettivi di ottimizzazione (numero di tagli trasversali) e non su tutti gli altri. Inoltre, vi incide in modo marginale e abbiamo quindi voluto ometterlo dal calcolo teorico, anche se, come per tutte le "variabili addizionali" nella programmazione pratica va comunque sempre considerato. Riassumendo, sia le "variabili addizionali" che i diversi ordinamenti delle combinazioni coil, possono essere trascurate nel calcolo teorico e non sono determinanti nell'influenzare la qualità dei risultati ottenibili con i diversi approcci risolutivi pratici.

2) La seconda considerazione riguarda i metodi tradizionali, le euristiche, le regole di buon senso che vengono seguite nelle diverse versioni ed interpretazioni dagli stessi programmatori durante la costruzione dei programmi. Abbiamo già citato nella precedente sezione, un tipico approccio che consiste nel partire direttamente da abbinamenti di larghezze con basso sfrido per poi passare ad abbinare i coil in funzione delle quantità richieste (pesi). Ve ne sono in realtà parecchie di queste strategie e tutte hanno lo scopo di arrivare ad una programmazione accettabile in un tempo ragionevole. Purtroppo questo approccio pragmatico, se risolve in qualche modo il problema di arrivare comunque ad una programmazione, non garantisce, come già detto, il perseguimento di tutte le "buone" opportunità e conduce anzi ad una parziale, ma sistematica perdita di efficacia ed efficienza che si traduce in mancato servizio ed erosione della marginalità.

3) La terza considerazione riguarda la relativamente recente possibilità offerta dalla tecnologia di adottare un nuovo metodo. Si tratta di adottare una nuova tecnica che comincia a diffondersi, che è stata sperimentata con successo e già adottata in produzione presso alcuni importanti centri servizio. Questo nuovo orientamento consiste nell'applicazione di un metodo sviluppato specificatamente per questa problematica negli ultimi dieci anni e che ha introdotto un approccio assolutamente innovativo ed efficace. È l'approccio che illustreremo brevemente nella prossima sessione.

IL NUOVO APPROCCIO, TECNICA RISOLUTIVA DIMENSIONAL UNFOLDING

Nelle precedenti sessioni, analizzando la complessità della problematica di programmazione abbiamo toccato con mano le ragioni per cui gli approcci tradizionali trovano tutti questi ostacoli e non riescono a garantire in modo sistematico una reale ottimizzazione della pianificazione. Abbiamo altresì appurato come una esplorazione a tappeto di tutte le possibilità sia di fatto impraticabile visto il numero enorme di combinazioni possibili pur considerando le "riduzioni" derivanti dalle cosiddette "variabili addizionali".

Tuttavia, è ad oggi disponibile un nuovo approccio già dimostratosi nella pratica particolarmente efficace. Negli ultimi anni infatti, è stata messa a punto una tecnica studiata apposta per la problematica di programmazione taglio ed implementata in un software già in uso in diverse realtà industriali che consente di ottenere risultati ritenuti dagli stessi utilizzatori molto efficaci. Questa tecnica consente di esplorare in modo "implicito" tutte le combinazioni interessanti (ossia senza la necessità di generare proprio tutte le possibilità, ma con la certezza di non perdere le migliori) garantendo quindi risultati ottimali in tempi rapidi.

Tale metodologia definita di "dimensional unfolding" consente di lavorare scomponendo le due dimensioni di ricerca delle soluzioni possibili (dimensione peso e dimensione altezza) e lavorando separatamente, ma contemporaneamente nelle due dimensioni. Questa tecnica di decomposizione in due dimensioni ha portato notevoli benefici in quanto:

– la complessità di questo nuovo approccio risulta essere di parecchio inferiore a quella teorica calcolata nella sessione 2. Infatti si passa da un numero teorico di combinazioni pari a: $P = M\,N = 2^{km} * 2^{n} = 2^{km+n}$. A un numero infinitamente piu' piccolo e maneggevole di combinazioni pari a: $P = (s\,M) + (r\,N) = s\,2^{km} + r\,2^{n}$.

Stiamo parlando di numeri ancora molto alti, ma trattabili con le attuali potenze di calcolo (nel primo caso n ed m si sommano all'esponente, mentre nel secondo caso si sommano i risultati delle loro potenze. Mentre s ed r sono due costanti che dipendono da alcune caratteristiche medie dimensionali di ordini e coil).

L'introduzione dei criteri di "dominanza" negli algoritmi di ricerca ha consentito di trattare in modo semplice e realistico la problematica degli obiettivi contrastanti. Nella generazione delle soluzioni vengono infatti sempre preservati tutti i programmi considerati "dominanti" secondo i diversi obiettivi di ottimizzazione definiti parametricamente e vengono scartati quelli "surclassati". Esiste quindi una semplice e intuitiva corrispondenza tra programmi "dominanti" e programmi "buoni".

La scelta finale tra i programmi "buoni" o "dominanti" puo' essere o automatizzata o lasciata all'operatore ed in entrambi i casi effettuata con criteri opportunistici legati al contesto della specifica sessione di lavoro (ad es. privilegiare il servizio, privilegiare riduzione cambi...).

Come già citato, questa nuova tecnica di "dimensional unfolding" è stata implementata in uno strumento software che supporta il programmatore "amplificandone" le capacità e potenziando ulteriormente la sua expertise. In pratica gli consente di esplorare automaticamente tutte le strade "interessanti" e gli garantisce quindi di non perdere opportunità. Si raggiunge in tal modo l'obiettivo iniziale di evitare quella parziale, ma sistematica emorragia di efficacia ed efficienza nella gestione di un processo aziendale strategico come la programmazione. Un ulteriore risultato incoraggiante è la capacità di ottenere ottimi risultati con lo strumento anche per i meno esperti e la drastica riduzione della *learning curve* di un programmatore.

Durante le diverse implementazioni già portate a termine lo strumento è stato altresì arricchito in modo da trattare tutti i tipici vincoli che si incontrano normalmente nella problematica di programmazione quali: diametro esterno min/max, peso singolo nastro, nastro centrale, invertibilità fogli, tolleranze di peso, vincoli delle macchine di taglio, quali massimo numero di lame, larghezza min/max, spessore min/max, diametro coil min/max, peso min/max in ingresso e in uscita, lunghezza foglio min/max, date consegna...

Il prodotto software che realizza questa tecnica è CCO (Coil Cut Optimizer) ed è stato realizzato da Fast-Square (www.fast-square.com), una società italiana che sta commercializzando e implementando il prodotto in Italia e che ha anche iniziato da quest'anno a presentarlo sul mercato estero. Tra i Centri Servizi italiani che già stanno utilizzando il prodotto in produzione vi sono S. Polo Lamiere, Comal-Ferlatta, Predieri Metalli. ■

BIBLIOGRAFIA

[1] E. Bertolotti, E. Castaldo, G. Giannone, *Near Optimal Objects packing through Dimensional Unfolding.* IAAI - Conference of Innovative Application of Artificial Intelligence Portland (USA), 1996.

[2] J.S. Ferreira. *A two-phase Roll Cutting Problem*, "European Journal of Operational Research" n.44, 1990, pp.185-196.

[3] S. Martello, P. Toth, *0/1 Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Son Ltd., 1990.

[4] J. Pearl, *Heuristics, Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, 1985.

[5] V. Pesce, *Ottimizzare i Programmi e le Linee di Taglio*, "Lamiera", aprile 2007, pp.46.

[6] M.G. Scutella, E. Bertolotti, E. Castaldo, M. Gambale, *Cutting Stock: Hypergraphs and Decomposition techniques*, Dip. Di Informatica, Universita' di Pisa. Proceedings of the Conference: Models and Algorithms, 1995.