Alessandro Agnetis

M O D E L L I C O M B I N A T O R I N E L L A P R O D U Z I O N E F L E S S I B I L E

Dispense a uso esclusivo degli studenti del corso di
Automazione Industriale

INDICE

INTRODUZIONE
PRODUZIONE, FABBRICA E MODELLISTICA

Un po' di storia

Con il termine *produzione* si intende il processo consistente nel trasformare materie prime in prodotti finiti. Questa definizione è valida per una larga classe di processi produttivi, siano essi chimici, lavorazioni meccaniche, assiematura. Un gran numero di processi manufatturieri riguardano la lavorazione dei metalli; spesso si tratta di processi di taglio. Alcune operazioni di taglio consistono nel generare superfici di tipo rotazionale e sono generalmente effettuate su torni, altre sono operazioni di intaglio e sono spesso effettuate da frese, altre sono operazioni di rifinitura e non asportano una grande quantità di metallo, e sono effettuate da mole. In ogni caso, queste varie macchine sono indicate col termine di *macchine utensili*, la cui evoluzione ha prodotto i Sistemi Flessibili di Lavorazione, che costituiscono l'oggetto di studio di questo testo.

Dal punto di vista storico, si può vedere che dalla fabbricazione di cannoni (16º secolo) alla metà del nostro secolo, la tecnologia delle macchine utensili si è sviluppata essenzialmente nel senso di un raffinamento della precisione e delle tolleranze. E' interessante notare che nel 1776 James Watt, inventore della macchina a vapore, era estremamente entusiasta dei cilindri fabbricati per lui da John Wilkinson, che erano "non più lontani dalla verità assoluta dello spessore di una moneta da sei pence nel punto peggiore..." (Lilley 1965).

Quando non è stato più un problema ottenere delle tolleranze accettabili, allora si è cominciato ad analizzare il problema di incrementare la produttività delle macchine. Inoltre, all'aumentare del volume produttivo richiesto dal mercato, la possibilità di riconfigurare rapidamente le macchine divenne via via più importante. Infatti, aumentando il numero di operazioni che una macchina è in grado di eseguire, ogni pezzo prodotto avrebbe dovuto visitare un numero inferiore di macchine, dando così luogo a un numero maggiore di pezzi prodotti per unità di tempo. Inoltre, la variabilità della domanda produttiva, sia in termini quantitativi che qualitativi, ha incoraggiato lo sviluppo di macchine in grado di effettuare diversi tipi di operazioni e in grado di riconfigurarsi – ossia di cambiare il tipo di operazione – in tempi sempre più ridotti: in altre parole, è andata acquisendo sempre maggiore importanza la *flessibilità* della macchina.

Le macchine di perforazione a più utensili, create verso la metà dell'800 (Gilbert 1980), e le prime macchine utensili semi–automatiche permisero di eseguire diverse sequenze di operazioni per mezzo di camme e altri meccanismi. Ma i tempi di set–up di queste macchine erano ancora elevati, e gli attrezzaggi usati erano complicati.

All'inizio del nostro secolo i progressi nella tecnologia dei materiali hanno reso possibile l'allungamento della vita media degli utensili e quindi tempi più lunghi tra due cambi–utensile. In termini di tipologia produttiva, una svolta fu rappresentata dall'introduzione dei primi componenti intercambiabili, e ciò aprì le porte alla produzione di *massa*, per la quale erano richieste macchine utensili in grado di garantire elevata velocità e buona precisione. La produzione di articoli complicati ciascuno dei quali richiedeva operazioni su macchine diverse diveniva via via più comune.

Oltre ai progressi nelle tecnologie delle macchine utensili, crebbe l'importanza della *divisione dei compiti*. Già nel 1776 Adam Smith aveva teorizzato la necessità di specializzare il lavoro di ciascun operaio per aumentare la produttività. I vantaggi nella divisione del lavoro possono ravvisarsi essenzialmente in: 1) rapidità di apprendimento da parte del lavoratore, trattandosi di un compito semplice e specifico, 2) risparmio dei tempi di set-up (grazie al fatto che ogni lavoratore esegue sempre le stesse operazioni); 3) possibilità da parte dei lavoratori di usare utensili e macchine per eseguire specifiche operazioni.

Le idee di Smith portarono alla *catena di montaggio*, che consiste nel disporre in serie stazioni di lavoro che devono eseguire operazioni in sequenza su ogni dato pezzo. Ogni stazione di lavoro (che originariamente consisteva di un essere umano) era preposta svolgere un compito estremamente semplice e specializzato. Dopo 130 anni dalla loro originale enunciazione, le idee di Smith furono applicate su grande scala da Henry Ford, che sviluppò una linea di assiematura per produrre un motore di automobile in 84 stadi. Ciò portò ad una significativa riduzione nei tempi di produzione e quindi a un abbassamento nei costi di produzione, e consacrò la catena di montaggio come struttura produttiva fondamentale per la produzione di massa. F. W. Taylor fu il primo, nel 1917, ad effettuare un'analisi scientifica della divisione del lavoro tra gli operai; il suo può quindi considerarsi come il primo contributo alla modellistica per la gestione dei sistemi industriali. Col progredire delle tecnologie, l'organizzazione seriale delle lavorazioni fu estesa a linee costituite da macchine automatiche (*linee di flusso* o *pipelines*). I pezzi entrano nel sistema grezzi, e visitano in sequenza le macchine che costituiscono la linea; ogni macchina esegue su ogni

pezzo una o più operazioni, al termine delle quali il pezzo si sposta dalla macchina su cui è montato alla macchina successiva.

Mentre le linee di flusso hanno consentito lo sviluppo della produzione di massa, l'aumento della domanda di prodotti sempre più specializzati e differenziati ha reso necessario il progetto di metodologie e architetture produttive su scala medio–bassa, senza perdere i vantaggi della produzione su larga scala. Ciò ha portato alla nascita del *job shop*, ossia un sistema di lavorazione in cui macchine dello stesso tipo sono dislocate fisicamente vicine, dividendo così l'officina in isole di macchine simili. Di conseguenza, si è resa necessaria una metodologia per raggruppare macchine e pezzi basandosi sulle similarità tra i processi produttivi dei vari prodotti, e questo ha condotto al concetto di *Group Technology*, verso la fine degli anni '50. L'ulteriore sviluppo di questo concetto ha determinato la nascita della *produzione cellulare*, in cui le isole sono in realtà delle *celle* indipendenti per quel che concerne pianificazione, controllo e amministrazione.

Negli anni '60 l'avvento dei calcolatori cominciò a influenzare fortemente la struttura e il funzionamento delle macchine utensili. La possibilità di tradurre il processo di lavorazione di un pezzo meccanico (*process plan*) in un programma automatico prefigurò l'opportunità di un consistente abbassamento nei tempi di sviluppo nei processi produttivi stessi, elevandone allo stesso tempo il grado di standardizzazione. Le prime macchine di questo tipo usavano un nastro perforato come supporto, e vennero chiamate *macchine a controllo numerico*. Tuttora viene usato questo termine per indicare, genericamente, macchine utensili programmabili. Data la loro versatilità, alcuni considerano le macchine a controllo numerico come la prima generazione di sistemi flessibili di lavorazione (Jaikumar e Van Wassenhove 1989, Bolwijn et al. 1986).

Col progredire della tecnologia dei calcolatori, anche insiemi di compiti di supervisione precedentemente assolti da uomini sono stati via via delegati al computer: oltre al controllo del movimento degli utensili, anche lo scambio degli utensili tra magazzino e mandrino è stato automatizzato, nonché il monitoraggio della vita degli utensili, o della loro rottura. In altre parole, è andata prendendo corpo la possibilità di un funzionamento completamente automatico delle macchine utensili. Una sola macchina a controllo numerico controllata da un computer (CNC) è in grado di eseguire una quantità di operazioni che prima richiedevano diverse macchine. Queste macchine possono identificarsi come la seconda generazione di sistemi flessibili di lavorazione.

Ciò che ancora mancava alle macchine a controllo numerico per diventare ciò che noi indichiamo genericamente con Sistema Flessibile di Produzione, era la loro interconnessione tramite un sistema di trasporto *integrato*, ossia supervisionato da un calcolatore centrale che gestisce e sincronizza i movimenti del sistema di trasporto, tenendo presente la situazione complessiva istante per istante dell'intero sistema. Ad esempio, la conoscenza di situazioni di congestione in determinate zone del sistema può consentire di reinstradare i pezzi lungo un altro itinerario.

Attività modellistica e attività algoritmica

Se dal punto di vista strettamente tecnologico gli FMS sono senza dubbio una pietra miliare, il loro impatto commerciale è stato, all'inizio, fortemente problematico: la loro introduzione nelle fabbriche non ha portato l'incremento di produttività (e quindi il ritorno di investimento) che ci si aspettava. In alcuni casi, la produttività addirittura è diminuita (Lewis 1981) (§1.4). Infatti, se è vero che un FMS di per sé costituisce un'apparecchiatura complessa, efficiente e dalle grandi potenzialità, è altrettanto vero che, in molti casi, è stata dedicata insufficiente attenzione ai problemi organizzativi e gestionali posti dall'introduzione e dal coordinamento di tali macchine. Ad esempio, per sfruttare appieno la versatilità operativa delle macchine, è necessario pianificare le attività delle varie stazioni di lavoro e del sistema di trasporto dei materiali in modo da evitare la creazione di inutili colli di bottiglia, di attese eccessive per i pezzi o il sovraffollamento dei magazzini intermedi. In altre parole, con il progredire delle tecnologie meccaniche e informatiche, la produttività potenziale di un FMS è andata crescendo; i tempi di cambio utensile (set up) si sono accorciati mentre la velocità di lavorazione è andata crescendo, senza andare a scapito della precisione. Allo stesso tempo, però, è cresciuta l'esigenza di sviluppare un corpo metodologico per *formulare* e *risolvere* un vasto insieme di problemi decisionali che prima non esistevano. A questi due passi fondamentali corrispondono due tipi di attività concettuali: la prima, *modellistica*, consistente nel rappresentare il sistema produttivo e il problema decisionale con strumenti matematici opportuni (§1.7); la seconda, *algoritmica*, che consiste nel trovare il modo più conveniente (non sempre quello ottimo) di risolvere un problema decisionale.

Il contributo di questo testo

In questo testo sono stati affrontati un insieme di problemi decisionali relativi alla pianificazione e alla gestione dei flussi materiali nei sistemi flessibili di lavorazione. Esiste ormai una letteratura abbastanza consolidata sulla nomenclatura e classificazione dei problemi decisionali nei sistemi flessibili (§1.6), ed una – meno consolidata – sulla classificazione strutturale dei sistemi flessibili (§1.8). In questo

testo si sono analizzati problemi decisionali analoghi in sistemi flessibili di tre diverse tipologie, alle quali è possibile ricondurre la grande maggioranza dei sistemi esistenti nel mondo (Jaikumar e Van Wassenhove 1989). L'obiettivo è stato quello di mettere in luce come le differenze strutturali e/o operative dei diversi sistemi si riflettono sulla struttura e sulla complessità dei modelli. Come vedremo infatti, le diverse modalità di interazione tra le risorse del sistema, nei vari casi danno luogo a modelli molto diversi. Per ogni particolare problema decisionale viene proposto un modello di tipo *combinatorio*, come conseguenza dell'intrinseca discretezza delle decisioni che nascono nel contesto tipico dei sistemi flessibili.

Rispetto ai modelli esistenti in letteratura (che spesso sono formulazioni di programmazione intera di medio–grandi dimensioni), i modelli presentati in questo testo sono concepiti con l'obiettivo di caratterizzare la *complessità* dei problemi relativi, cercando di isolare i casi semplici (*polinomiali*) da quelli difficili (*NP–completi*), possibilmente riconoscendo strutture più particolari all'interno dei singoli modelli decisionali. Così vedremo ad esempio che certi problemi di assegnamento di operazioni a macchine possono essere ricondotti a problemi di percorso minimo (§§2.4, 2.5); o che in alcuni problemi di sequenziamento è utile ottenere dei limiti inferiori sulla soluzione ottima risolvendo un problema di trasporti (§3.3). In alcuni casi, la frontiera tra problemi semplici e difficili si riesce a tracciare in modo particolarmente netto, individuando quali elementi del problema lo collocano in una categoria o l'altra, come nel caso dei problemi di assegnamento degli utensili nelle celle flessibili (§2.4.4).

Nel §1 viene presentato l'inquadramento concettuale del lavoro svolto, sia dal punto di vista applicativo che da quello metodologico. Dal punto di vista applicativo, vedremo alcune delle problematiche di tipo modellistico poste dall'avvento della cosiddetta fabbrica automatica. Dopo una breve discussione sull'integrazione manifatturiera (§§1.1 e 1.2), introdurremo i "componenti fondamentali" della fabbrica che interessano la nostra analisi (§1.3); seguirà una descrizione più dettagliata degli FMS e una breve discussione sulla riuscita (o meno) della loro implementazione (§1.4). Quindi ci soffermeremo sul concetto di flessibilità, proponendone una nuova caratterizzazione (§1.5), e illustreremo qualitativamente i problemi decisionali che nascono nell'ambito dei sistemi flessibili (§1.6). Da un punto di vista metodologico, vedremo una breve discussione comparata degli approcci modellistici usati in letteratura, chiarendo il ruolo dell'ottimizzazione combinatoria rispetto ad altri strumenti matematici (simulazione, modelli analitici etc.) (§1.7). Infine, ci soffermeremo sulla classificazione e caratterizzazione dei vari tipi di sistemi flessibili, sulla base della quale è strutturato il resto del testo (§1.8).

Successivamente, vengono illustrati modelli e algoritmi per i problemi decisionali relativi alla gestione dei flussi fisici in tre classi di sistemi flessibili, distinguendo in particolare le *celle flessibili* (§2) dai *sistemi pipeline* (§3) e dai *sistemi flessibili* propriamente detti (§4).

1. LA FABBRICA AUTOMATICA

1.1. L'integrazione manifatturiera

Negli ultimi anni, lo sviluppo delle nuove tecnologie — e tra queste va inclusa anche il corpo metodologico usualmente indicato come "scienza dell'organizzazione e della gestione" — ha inciso profondamente sulla struttura e sul funzionamento dei sistemi produttivi. Tale mutamento è soprattutto visibile per quel che concerne l'industria manufatturiera. Infatti, le tecnologie elettroniche, informatiche e meccaniche e la loro integrazione (come nel caso della robotica) hanno reso possibile l'automazione di molti processi produttivi, nonché il conseguimento di un gran numero di benefici quali, ad esempio, minori costi di produzione, maggiore qualità del prodotto, e un più elevato grado di *flessibilità* produttiva. Il concetto di flessibilità può essere inteso in molti modi, come vedremo più avanti (§1.5); in ogni caso, possiamo dire che tanto più un sistema produttivo è flessibile, tanto più è in grado di adattarsi rapidamente alla variabilità del mercato, ossia di mutare quantità e tipologia produttiva, inseguendo la domanda.

Di pari passo con lo sviluppo tecnologico, si è capito che per utilizzare in modo ottimale i sistemi automatici (computer, robot…), è necessario lo sviluppo di un nuovo approccio organizzativo, di un'attitudine ad affrontare i problemi in modo globale e "integrato", e, in definitiva, di nuovi metodi di *rappresentare il sistema fabbrica*.

Infatti, come accennato nell'introduzione, l'approccio organizzativo mutuato dal Taylorismo prevedeva una suddivisione del processo produttivo in fasi distinte, ciascuna delle quali richiede, all'operatore umano incaricato di eseguirla, un elevato grado di specializzazione. Questa filosofia organizzativa nacque in un periodo storico in cui la produzione manufatturiera era sostanzialmente omogenea (limitato numero di prodotti diversi) e stazionaria (la domanda non mutava con rapidità ed era quindi relativamente prevedibile). Di conseguenza, era logico organizzare la produzione in modo da massimizzare l'efficienza, anche a costo di avere elevati tempi di riconversione delle risorse – sia materiali che umane – qualora si verificassero mutamenti nella tipologia produttiva. Inoltre, in simili condizioni, risultava anche limitata la quantità di informazioni che i diversi reparti si dovevano scambiare per portare avanti il processo produttivo stesso.

Con l'andare del tempo, l'aumento e la diversificazione dei prodotti richiesti dal mercato e dei processi disponibili al sistema produttivo hanno rivelato l'inefficenza di questo tipo estremamente rigido di organizzazione, portando verso la strutturazione

dell'azienda in aree funzionali, ognuna preposta allo svolgimento di un certo tipo di compito. Nell'ultimo decennio, sono andate assumendo sempre maggiore importanza la capacità del sistema di *riconfigurarsi* per far fronte alle nuove esigenze del mercato, e la necessità di rendere efficiente lo scambio di informazioni tra i reparti della fabbrica durante il ciclo di vita di un progetto e di un prodotto, suggerendo così di rivedere la suddivisione dei compiti tra le varie funzioni aziendali. Basti pensare, ad esempio, alle attività connesse alla gestione di un ordine riguardante un prodotto che differisce leggermente da uno già presente nel "catalogo" dell'azienda: secondo l'approccio tradizionale, ovvero "non integrato", il nuovo prodotto vedrà la luce dopo un complicato scambio di informazioni (ognuno dei quali avviene perlopiù a mezzo di moduli cartacei, e dunque con dispendio di tempo e scarsa affidabilità) tra i reparti vendite, progettazione, ingegnerizzazione, costi e produzione: infatti, l'ordine del reparto vendite viene passato al reparto progettazione, che a sua volta interpellerà i tecnici dell'ingegnerizzazione per sapere se esistono già progetti analoghi; questi a loro volta interagiranno con il reparto produzione per quel che riguarda lo sviluppo dei programmi a controllo numerico relativi. Secondo la filosofia della integrazione delle funzioni aziendali, invece, lo scambio di informazioni è limitato dalla presenza di un unico sistema informativo aziendale, a cui tutti i reparti possono accedere. Ma un aspetto ancora più sostanziale della filosofia *CIM* (*Computer Integrated Manufacturing*), sta nella tendenza ad aggregare quelle aree funzionali che richiedono una elevata interazione per svolgere il loro lavoro, eliminando così la stessa necessità legata allo scambio di informazioni (Scheer 1988). Infatti, il principale motivo per specializzare capacità e competenze (gli *skill*) stava nelle limitate capacità elaborative degli individui; ora che una parte del lavoro può essere demandata ai calcolatori, e che la comunicazione con essi avviene tramite interfacce "amichevoli", vengono a cadere molti presupposti per separare certe attività. Così, ad esempio, la progettazione può direttamente fornire l'input per le macchine a controllo numerico che dovranno produrre quell'oggetto (integrazione CAD/CAM); la conoscenza delle risorse (fisiche e/o informative) disponibili ad un certo istante dall'azienda può essere utilizzata per decidere il processo più conveniente per produrre certi prodotti (integrazione tra ingegnerizzazione, gestione scorte, ufficio costi…), e così via.

1.2 Risorse, flussi informativi e modellistica del sistema azienda

L'evoluzione della concezione e della filosofia organizzativa produttiva consente di rappresentare la fabbrica ed i flussi (informativi e materiali) che scorrono in essa per mezzo dello schema presentato in Fig.1.1. Nella fabbrica è possibile distinguere un insieme, estremamente vasto ed eterogeneo, di *risorse*, intendendo con questo termine

tutto ciò di cui si ha bisogno per portare avanti il processo produttivo, e che interviene in esso subendo o producendo trasformazioni. Avremo dunque vari tipi di risorse:

– *fisiche*: esse includono ovviamente materie prime, componenti, semilavorati, sottoassiemi, macchinari, calcolatori, utensili, sistemi di movimentazione, magazzini, energia, prodotti finiti,…;

– *informative*: sono le più svariate, e vanno dalle specifiche tecniche necessarie per eseguire particolari operazioni di taglio, agli skill relativi a una metodologia progettuale, alle proiezioni sull'andamento della domanda in un particolare segmento di mercato…;

– *finanziarie*: disponibilità liquida o capitale immobilizzato;

– *umane*: quantità e qualità del personale nelle diverse specializzazioni e fasce di inquadramento;

– *temporali*: il tempo è infatti anch'esso una risorsa, spesso anzi è una risorsa critica, sia che si tratti del tempo per eseguire una particolare lavorazione, sia di quello per ottenere una fornitura, sia del tempo di consegna richiesto dal cliente.

Il processo produttivo consiste di un insieme di *azioni* che operano su queste risorse. Ad esempio, la progettazione di un pezzo utilizza risorse di tipo informativo (l'archivio progetti, le metodologie e i criteri di progettazione), fisico (le workstation CAD), umano; una serie di lavorazioni meccaniche su di un grezzo è un'attività che impiega, ancora, risorse di tipo informativo (i programmi per le macchine a controllo numerico, i programmi di supervisione, la conoscenza dello stato dei macchinari…), fisico (la macchina a controllo numerico, il pezzo che deve essere lavorato, il sistema di trasporto, i magazzini…), eventualmente umano etc. Anche l'insieme delle azioni è ovviamente molto vasto, e comprende: progettazione, ingegnerizzazione, lavorazioni, produzione, ricerca e sviluppo, istruzione, allocazione di risorse, monitoraggio, …. Peraltro, le attività non sono necessariamente in corrispondenza biunivoca con uffici o reparti dell'azienda.

Le azioni possono *consumare*, *usare* o *trasformare* risorse. Ad esempio, si consideri un'operazione di taglio su di un pezzo: essa *consuma* una certa quantità di tempo e di energia, che scompaiono per effetto dell'azione stessa; *usa* macchine, utensili ed eventualmente operai, che dopo l'operazione sono nuovamente disponibili per altre azioni; *trasforma* il semilavorato in un altro, più "vicino" al prodotto finito. O ancora, un corso di aggiornamento *consuma* tempo, *usa* insegnanti e materiale didattico e *trasforma* gli utenti in individui con un parco di conoscenze più ricco. Se

indichiamo con R l'insieme di tutte le risorse, si possono allora vedere le azioni come un'applicazione dall'insieme 2^R delle parti di R, allo stesso insieme 2^R: infatti, ogni azione opera su un sottoinsieme di risorse, e ne restituisce un altro sottoinsieme.

L'interazione tra le risorse determinata dalle azioni viene resa possibile da un *flusso informativo* bidirezionale (Fig.1.1), il quale in un verso registra e misura lo stato attuale delle risorse del sistema; nell'altro va ad influenzare tale stato, attraverso il rilascio di documenti (ordini di produzione, di progettazione, di manutenzione…) che realizzano le azioni stesse. In altri termini, è fondamentale considerare l'esistenza, parallelamente alla realtà fisica, di una "immagine informativa" dell'azienda, che registra e influenza l'evoluzione dell'azienda stessa.

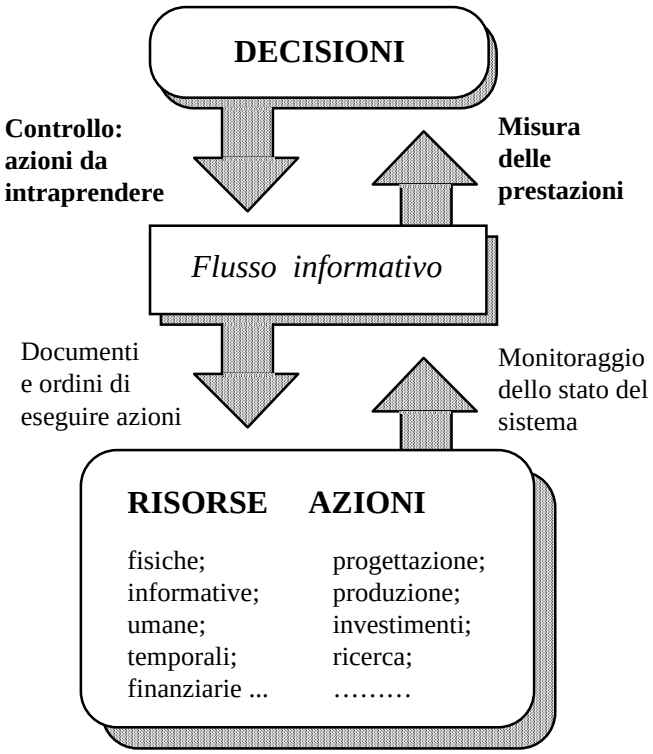


Figura 1.1. Schema concettuale dell'azienda.

Tale insieme di informazioni costituisce l'interfaccia tra la realtà "fisica" dell'azienda e il *livello decisionale*, ossia quell'insieme di attività che determinano, appunto, le azioni da intraprendere. Infatti, quale che sia la filosofia o i princìpi che guidano i vari tipi di decisioni, esse vengono prese sulla base di una conoscenza del sistema, che si ha attraverso una sua *rappresentazione*: a questo livello, dunque, si colloca l'attività modellistica, che deve fornire, di volta in volta, la "chiave di lettura" più opportuna per un certo tipo di decisione, ossia isolare le grandezze significative, descriverne i rapporti di causalità, porsi al corretto livello di aggregazione. L'attività decisionale richiede dunque la conoscenza di alcune grandezze rappresentative dello

stato del sistema, e fornisce le decisioni in termini di azioni di controllo, che la struttura informativa dovrà quindi tradurre in opportuni documenti e ordini di azioni.

E' chiaro che la consistenza tra lo stato attuale del sistema e i dati che lo descrivono è un fattore critico affinché le decisioni prese possano rivelarsi efficaci. Infatti, tanto per fare un esempio, una procedura (ovvero un modello) che decide se, come e quando effettuare degli ordini di certe scorte, basa tale decisione sullo stato attuale del magazzino, ovvero sulla quantità di ogni tipo di articolo presente nel magazzino all'istante attuale. E' dunque necessario che i dati relativi al magazzino noti alla struttura informativa corrispondano alla realtà, in quanto è sempre su tali dati che "agiscono" i modelli decisionali e di controllo.

In Fig.1.1 sono riportati in grassetto gli aspetti che più specificamente costituiscono l'oggetto di questo testo.

Possiamo classificare i tipi di decisione secondo vari criteri. Uno di essi è quello basato sulla lunghezza dell'orizzonte temporale nel quale le decisioni da prendere faranno sentire il loro effetto. In modo non rigido, possiamo identificare così alcune classi di problemi decisionali: problemi *strategici* (ovvero, in cui va determinata la struttura e la composizione dell'azienda, e riguardano perciò il lungo termine), *tattici* (di pianificazione a medio-breve termine), *operativi* (di gestione dei flussi nel breve termine o in tempo reale).

In questo testo parleremo più in dettaglio dei problemi decisionali riguardanti la pianificazione di medio e breve termine. Prima è però necessario fare un'osservazione di carattere generale, relativa all'impiego dei modelli nei sistemi manifatturieri.

E' evidente che l'idea di concepire un modello del sistema che, da solo, riesca a inglobarne tutti gli aspetti decisionali, è del tutto impraticabile. Ciò non solo a causa della complessità del compito, ma anche e soprattutto riflettendo sulla natura intrinsecamente distribuita, ossia multidecisore, di un processo gestionale, con i conseguenti problemi di coordinamento tra i decisori, spesso risolti organizzando il sistema in modo gerarchico. Un processo gestionale gerarchico comporta, per esempio, decisioni che si ripercuotono sul sistema produttivo a livelli differenti rispondendo a esigenze mutevoli nel tempo con andamento in genere tanto meno noto quanto più esteso è l'intervallo di pianificazione. Si parla a questo proposito, relativamente ai livelli tattico e strategico, di pianificazione con *orizzonte sfumato*, cioè tanto meno definito quanto più le competenze del livello gerarchico tendono a sconfinare nelle competenze del livello immediatamente superiore. Ad esempio, è del tutto inutile pianificare con esattezza tutte le operazioni che le macchine del sistema

manifatturiero dovrebbero compiere nell'arco di un anno, in quanto nel giro di un periodo molto più breve le condizioni in cui si opera (gli ordini, le scorte, le stesse macchine disponibili etc.) risulteranno certamente modificate.

Quello che si cerca di fare è perciò modellare alcuni sottoproblemi decisionali, fissando quei parametri che dipendono da altre decisioni, e la cui esplicita considerazione comporterebbe un aumento della complessità del modello o dell'incertezza tale da renderlo inutilizzabile in pratica. Ad esempio, nei problemi di gestione dei flussi che vedremo più avanti, supporremo noto e immutabile il processo produttivo, ossia si suppone che la successione di operazioni che un certo pezzo deve subire per diventare prodotto finito sia un dato di ingresso del problema. Prevedendo invece di poter utilizzare processi produttivi diversi per pezzi anche dello stesso tipo, è in generale possibile, ad esempio, giungere ad una migliore ripartizione del carico di lavoro tra le macchine, che è uno degli obiettivi della gestione dei flussi. Tuttavia, considerare anche questo aspetto avrebbe impedito, ad esempio, la formulazione del problema di instradamento nel modo semplice ed efficiente presentato nel §4.4. In altri termini, anche se ciò non verrà detto sempre in modo esplicito, nel seguito faremo uso dei concetti di *decomposizione* e di *restrizione*. Ossia, si *decompone* un problema più grande in più sottoproblemi, perdendo la garanzia che la soluzione ricostruita dai singoli sottoproblemi corrisponda all'ottima globale, ma guadagnando in efficacia rappresentativa, isolando e caratterizzando quelle parti del processo decisionale che sono più "semplici" — ovvero che si sanno risolvere in modo più efficiente. Inoltre, fissando a priori alcuni aspetti della soluzione, si *restringe* la regione ammissibile, allo stesso scopo di poter poi risolvere il problema.

1.3. Architettura e componenti fondamentali di un sistema di produzione

In questo paragrafo daremo una descrizione schematica dell'architettura di un sistema di produzione e dei suoi componenti fondamentali, mostrando per ognuno il tipo di caratterizzazione che utilizzeremo nel corso del testo. Non tratteremo in dettaglio molti aspetti realizzativi–tecnologici, non essendo del resto questa la sede più appropriata per tale trattazione. Come dovrebbe tuttavia risultare chiaro nel seguito, il tipo di caratterizzazione adottato consente di ricondurre a esso una classe molto vasta di casi reali, con l'ulteriore vantaggio di potere almeno in parte modellare, così facendo, situazioni relative a settori anche molto diversi. Soprattutto, la caratterizzazione è già di per sé "orientata ai problemi", ossia a mettere in luce quegli aspetti dei componenti del sistema produttivo rilevanti ai fini dei problemi decisionali da risolvere (vedi §1.7.4).

Tra le risorse fisiche del sistema, possiamo individuare da un lato i componenti del sistema, quali le unità operatrici, il sistema di trasporto dei materiali, i magazzini, e dall'altro i materiali. Schematizzando, possiamo dire che un sistema di produzione è costituito dai seguenti elementi:

- *oggetti della produzione* (materiali grezzi, semilavorati, prodotti finiti);
- *unità operatrici* (nel caso di produzione meccanica: macchine utensili, centri di lavorazione, robot, stazioni di lavaggio, di prova etc.; in generale qualunque risorsa, anche umana, adibita alla lavorazione di prodotti);
- *risorse* e *strumenti* necessari alla lavorazione dei prodotti (utensili e simili);
- *sistema di trasporto* (nel caso di produzione meccanica: carrelli a guida automatica (AGV), rulliere, robot; in generale qualunque risorsa, anche umana, adibita alla movimentazione di pezzi, prodotti o utensili nel sistema);
- *magazzini* (nel caso di produzione meccanica: buffer, portapezzi (pallet), magazzini utensili, magazzini prodotti; in generale, contenitori più o meno specializzati, distribuiti nel sistema e collegati da un sistema di trasporto, adibiti alla temporanea conservazione di pezzi, prodotti o utensili);
- *sistema informativo* e di controllo (distribuito nel sistema e collegato da una rete informativa in genere di tipo gerarchico, adibita alla elaborazione delle informazioni relative alla produzione, del flusso dei documenti e al controllo del sistema nel suo complesso).

  Una possibile architettura di un sistema di produzione è rappresentata in Fig. 1.2.

Vediamo ora le risorse fisiche coinvolte, che, come abbiamo detto, oltre agli elementi del sistema includono anche i prodotti intesi come singoli pezzi in lavorazione.

1.3.1. Prodotti

Siccome siamo interessati a processi di tipo manufatturiero, il risultato della produzione è un insieme di parti discrete, distinte e ciascuna delle quali, in linea di principio, ha una sua individualità. In altre parole, i modelli sviluppati in questo testo non riguardano processi di tipo continuo, quali quelli che ad esempio si hanno nell'industria chimica o petrolifera. Ogni prodotto è costituito da un insieme di *componenti* che, in base ad un certo *processo produttivo* (*process plan*), vengono assiemati e lavorati in modo da dare luogo ad un certo *prodotto finito*. L'insieme dei componenti che concorrono a formare la singola copia di un prodotto – anche se

ancora non assiemati – prende il nome di *unità* di prodotto. Se il processo non è di assiematura, e dunque consiste di un insieme di operazioni che devono essere subite da un'unica entità fisica, parleremo spesso di *pezzo* anziché di unità. Un sottoinsieme di componenti già assiemati prende il nome di *sottoassieme* (*subassembly*).

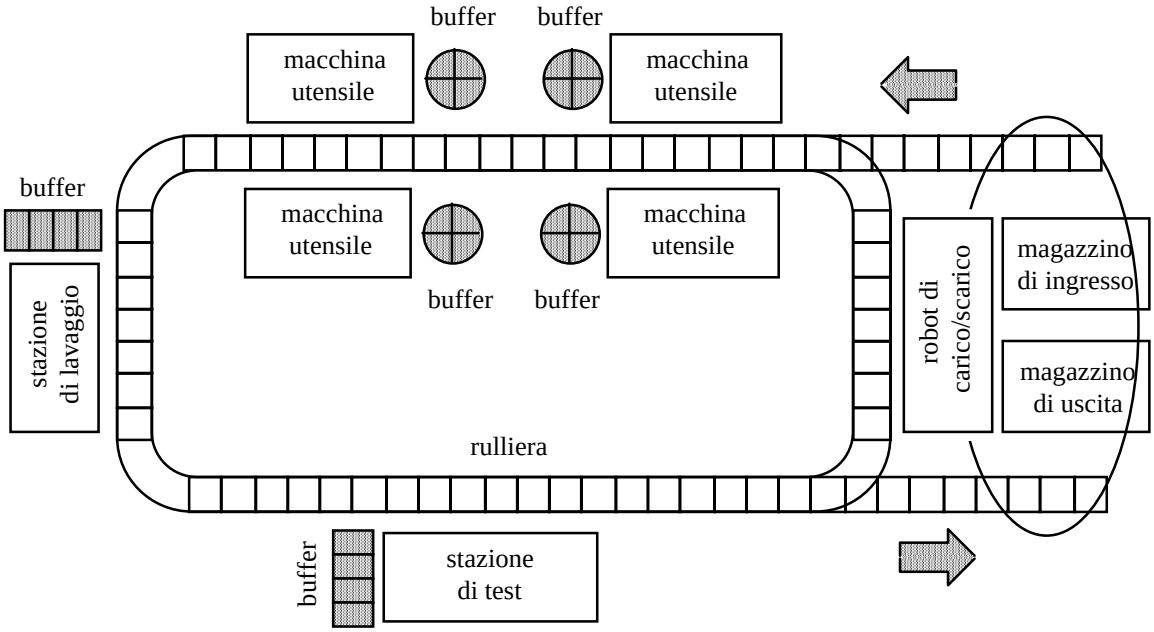


Figura 1.2. Schema di una possibile architettura di sistema (flessibile) di produzione.

La trasformazione dei componenti elementari in prodotti finiti avviene tramite un insieme di *operazioni* particolari che necessitano, per essere eseguite, di una *unità operatrice* e, in generale, di altre risorse, tra cui uno o più *utensili* (§1.3.4). Supporremo sempre che le operazioni non siano interrompibili.

Dal punto di vista tecnologico, i tipi di operazioni che ciascuna unità subisce possono essere, ovviamente, svariate; dal punto di vista concettuale, è importante distinguere tra due tipi sostanzialmente diversi di operazioni: quelle di *lavorazione* propriamente dette e quelle di *assiematura*. Le prime agiscono su un certo sottoassieme (sottoponendolo a trattamenti che possono essere di taglio, saldatura, tornitura etc); le seconde effettuano l'unione di due o più sottoassiemi (anche qui, attraverso procedimenti di tipo meccanico e/o chimico).

Ad ogni copia di un certo tipo di prodotto finito può essere dunque associato un *job*, consistente dell'insieme delle operazioni. In generale, le operazioni richieste per la produzione di un pezzo non possono essere svolte in ordine arbitrario, bensì è definita una *relazione di precedenza*, esprimibile in termini di un grafo G orientato e, ovviamente, aciclico (*grafo delle operazioni*). I nodi di questo grafo corrispondono alle operazioni, gli archi rappresentano invece rapporti di precedenza. Un caso di particolare interesse si ha quando G è un cammino, e in tal caso si parla anche di

catena delle operazioni. Un altro caso di grande importanza è quello in cui *G* è un albero, come è tipico dei processi di assiematura (*albero di assiematura*). Si noti che sul grafo delle operazioni, una operazione di assiematura corrisponde a un nodo avente grado d'ingresso maggiore di 1. In un albero di assiematura, la radice corrisponde al prodotto finito, pronto per essere inviato fuori; le foglie, invece, a operazioni sui componenti elementari.

Ciascuna operazione è inoltre caratterizzata da altre informazioni fondamentali. Anzitutto, ogni operazione richiede, per essere eseguita da ciascuna macchina, un certo tempo (*tempo operazione*). Tale tempo in generale può essere diverso asseconda di quale unità operatrice sarà incaricata di eseguire quella operazione, ma per semplicità espositiva supporremo che tale durata sia la stessa su tutte le unità operatrici in grado di svolgere l'operazione.

Un'altra informazione fondamentale associata alle operazioni riguarda le risorse produttive impegnate. Oltre all'impiego di una unità operatrice, infatti, tipicamente una operazione richiede l'uso di un *utensile* (vedi §1.3.4), ed eventualmente di altre risorse presenti nel sistema. Per poter effettuare l'operazione, l'unità operatrice deve essere dotata dell'utensile corrispondente.

Un ordine di produzione in genere consiste di un insieme di unità, ciascuna delle quali va prodotta in una certa quantità. Ossia, spesso le unità individuali da prodursi nel prossimo arco di tempo (giorno, settimana, mese…) sono raggruppabili in *tipi* (*part type*), ciascuno dei quali corrisponde a un determinato tipo di prodotto e richiede per ognuno dei suoi elementi il medesimo processo produttivo; questo fatto, come vedremo (§4.4) è spesso molto utile per risolvere efficientemente particolari problemi di decisione. Il numero dei pezzi di ciascun part type (*domanda di prodotto*) può, a seconda del tipo di produzione, variare da alcune unità a diverse centinaia. Inoltre, a parte dove indicato esplicitamente il contrario, supporremo sempre che tutti i componenti siano disponibili all'inizio del funzionamento del sistema (release dates nulle).

1.3.2. Sistema di trasporto

I modi in cui, in un sistema di produzione industriale, può essere realizzato il sistema di trasporto dei materiali e degli utensili varia a seconda delle esigenze produttive, della flessibilità di instradamento richiesta (§1.5.1), dei costi. I mezzi più comunemente utilizzati vanno dalle rulliere (economiche e affidabili, offrono un servizio di trasporto sempre disponibile ma sono in genere poco flessibili e piuttosto lente), ai carrelli a guida automatica (veloci e flessibili, ma presenti nel sistema in

quantità limitata, e inoltre più costosi e in genere meno affidabili delle rulliere), ai robot (il massimo in termini di flessibilità ma anche di costo).

Un elemento caratterizzante della flessibilità del sistema di trasporto dei pezzi in produzione è la sua *topologia*, rappresentabile per mezzo di un grafo $G_T = (M, E)$, che può essere orientato oppure no, a seconda della situazione modellata. In questo grafo i nodi corrispondono alle unità operatrici, gli archi rappresentano le connessioni dirette. Questo grafo può assumere forme diverse. Una struttura molto comune è quella in cui il grafo G_T è un cammino; in questo caso tutti i pezzi seguono lo stesso instradamento e visitano tutte le unità operatrici nello stesso ordine. In questo caso diciamo che la struttura del sistema è di tipo *pipeline* (§3). Quando la movimentazione dei materiali è svolta per mezzo di carrelli a guida automatica o robot, è possibile un maggior numero di collegamenti tra unità operatrici, e il grafo G_T è più denso. Se il layout consente la diretta connessione di ogni coppia di macchine, in particolare, il grafo G_T è completo.

1.3.3.Unità operatrici

Con il termine "unità operatrice" si intende qualunque risorsa in grado di eseguire operazioni sulle unità in produzione. A seconda dei casi, un'unità operatrice può essere identificata con un centro di lavorazione, un robot, una macchina utensile, un gruppo (o *pool*) di macchine, o anche, in un ambiente non automatizzato, con una o più unità di personale.

Una macchina può eseguire certe operazioni se è dotata degli utensili opportuni. Gli utensili vengono caricati, all'inizio di un certo periodo di lavorazione in un magazzino a bordo macchina (§1.3.4), e che tale configurazione non venga mutata fino a un successivo *set–up* del sistema. Infatti, l'accesso a tali magazzini comporta l'impiego di personale e di un certo tempo (proporzionale al numero di operazioni di inserzione/sostituzione). Tuttavia, in alcuni casi (§3.2), qualora la capacità dei magazzini e il costo di duplicazione degli utensili non rappresentino un fattore critico, può convenire considerare le macchine *general purpose*, ossia in grado di eseguire qualunque operazione, trascurando i problemi posti dall'attrezzaggio. Un caso in cui questo spesso accade è quello delle classiche linee di assiematura, in cui i compiti vengono svolti da persone — le unità operatrici più flessibili che si conoscano.

Un importante elemento decisionale associato a ciascuna unità operatrice è il carico di lavoro che essa assume in seguito all'assegnazione delle lavorazioni. Tale carico è in generale espresso in termini del tempo totale delle lavorazioni eseguite

sulla macchina. Molti indici di prestazione del sistema sono legati ai valori dei carichi di lavoro delle varie macchine.

1.3.4. Utensili, magazzini utensili e altre risorse

Nelle lavorazioni viene utilizzato un insieme *U* di utensili. Con questo termine indichiamo quell'insieme di risorse di cui deve essere dotata l'unità operatrice per poter effettuare una certa operazione su un'unità. Tipici utensili sono ad es. quelli di taglio, spesso molto numerosi ed estremamente sofisticati e costosi; ma con la stessa accezione si possono considerare "utensili" altre risorse, come quelle software, ossia ad esempio i microprogrammi di controllo di un robot (§2.4). In ogni caso, siamo interessati a considerare solo quegli utensili la cui limitata disponibilità pone particolari problemi di gestione — quindi non considereremo altre risorse, come quelle energetiche, essenziali ma abbondanti, almeno a livello dei problemi gestionali considerati in questo testo.

Ciascuna delle unità operatrici di un sistema è dotata di un *magazzino utensili* in grado di ospitare una quantità limitata di utensili. Il numero di posti (*slot*) disponibili in un magazzino utensili rapresenta la *capacità* del magazzino. I magazzini utensili localizzati presso le unità operatrici sono in genere del tipo "a catena", dotati normalmente di non più di un centinaio di slot.

Ogni tipo utensile è caratterizzato da un insieme di parametri; quello che prenderemo talora in considerazione esplicitamente è il *costo*. In un sistema di lavorazione, gli utensili di tipo diverso possono facilmente essere in numero alquanto elevato (anche parecchie centinaia).

Si noti che un'operazione può impiegare anche altre risorse del sistema, oltre a una unità operatrice e gli utensili. Ad esempio, il montaggio di particolari componenti su scheda richiede un robot per il posizionamento, eventuali utensili (ad es. cacciaviti), e un ripiano di lavoro, che fa parte del sistema di lavorazione. Quest'ultima risorsa è però concettualmente distinta dagli utensili in quanto non è assegnata in modo permanente alla macchina, ma viene da essa impiegata solo per l'intervallo di tempo necessario a compiere l'operazione, dopodiché viene rilasciata. Inoltre, talvolta è necessario considerare esplicitamente quelle risorse che servono a portare i pezzi nel sistema (*pallet*); se i pallet non sono in numero elevato ciò può limitare il numero complessivo di unità in corso di produzione.

Tuttavia, va detto che in alcuni casi, gli utensili sono in realtà assimilabili a risorse del sistema, in quanto possono essere dinamicamente allocati a diverse unità

operatrici. Questo è quanto accade in particolari architetture (§2.5), in cui gli utensili, durante il normale funzionamento del sistema, vengono prelevati e riposti da un *magazzino centrale*, concepito come uno scaffale (o rastrelliera) al quale ha accesso un sistema di "storage & retrieval" (spesso realizzato da un robot). In questo tipo di architettura, informazioni essenziali riguardano i tempi di spostamento dei dispositivi preposti allo spostamento degli utensili, nonché i tempi di accesso al magazzino utensili. I centri di lavorazione possono essere o meno dotati di un loro magazzino utensili locale.

1.3.5. Buffer

Il termine *buffer* è comunemente utilizzato in ambito manifatturiero per indicare il magazzino in cui vengono collocati provvisoriamente i pezzi in attesa di lavorazione presso un'unità operatrice. Tipiche soluzioni realizzative per un buffer sono quelle mostrate in Fig.1.3. La prima consiste di un supporto rotante su cui trovano posto i pezzi, ed ammette in genere modalità di gestione più flessibili della seconda (magazzino lineare), normalmente a gestione *FIFO*. In un sistema automatizzato, la funzione del buffer non è solamente quella di accogliere i pezzi in attesa, ma anche quella di presentarli alla macchina nella posizione e con le tolleranze previste per poter eseguire le operazioni con la precisione richiesta: ciò comporta un costo relativamente elevato, che fa sì che il numero di pezzi che un buffer è in grado di accogliere sia in genere limitato a poche unità (§3.4). La *capacità* del buffer è il numero di pezzi che esso è in grado di ospitare. Un discorso diverso vale per quei tipi di buffer non dedicato, cioè non associato a una particolare unità operatrice, che prendono il nome di *polmoni* (Fig.1.3.*b*) e sono utilizzati in molti sistemi per la gestione dei pezzi in ricircolo (dei pezzi cioè che non hanno potuto trovare posto in nessuno dei buffer dedicati).

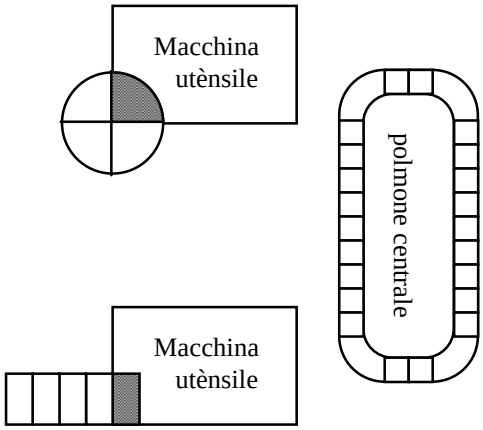


Figura 1.3. Magazzini semilavorati.

1.4. Sistemi Flessibili di Lavorazione (FMS) e di Assiematura (FAS)

Secondo una delle numerose definizioni che di Sistema Flessibile di Produzione (in seguito chiamato *FMS, Flexible Manufacturing System*) esistono in letteratura, un FMS è "un gruppo di stazioni di lavoro, di uso generale o dedicate, a controllo numerico o tradizionali, collegate fisicamente tra di loro mediante un sistema di trasporto automatizzato, che agiscono sotto il coordinamento di un sistema computerizzato di controllo. Ogni stazione deve essere dotata di un sistema automatico di carico e scarico, e, per quelle per cui ha senso, di cambio automatico dell'utensile. Il sistema deve poter accettare in ingresso pezzi differenti con arrivi casuali, e deve essere in grado di ottimizzare il flusso dei materiali al suo interno" (Manuelli et al. 1984). Il termine *FAS* (*Flexible Assembly System*) è riferito a quei sistemi in grado di svolgere anche operazioni di assiematura.

Questi sistemi sono progettati allo scopo di conservare i vantaggi della produzione su larga scala evitando gli inconvenienti tipici dei job shop, quali un elevato work in progress e lunghi tempi di set up.

Alcuni dei vantaggi riconosciuti per gli FMS riguardano:

— La capacità di produrre diversi tipi di prodotti;

— Elevata utilizzazione: siccome ogni macchina è flessibile, può essere impiegata per molte operazioni diverse e ciò conduce ad una elevata utilizzazione;

— Se è vero che il costo di ogni macchina è elevato, il numero di macchine necessario per eseguire un certo insieme di lavorazioni è inferiore rispetto al caso di macchine dedicate; di conseguenza lo spazio di lavoro globalmente occupato dalle macchine è inferiore;

— Costi di manodopera inferiori, come conseguenza dell'automazione della movimentazione di utensili e parti;

— Lead times inferiori: essendo flessibili, questi sistemi possono infatti riconfigurarsi velocemente;

— Ridondanza: qualora una macchina sia temporaneamente fuori uso, la versatilità delle macchine stesse fa sì che le stesse operazioni possano – almeno in gran parte – essere ridistribuite tra le altre macchine;

— Modularità: è possibile introdurre le macchine in tempi successivi, adattando gradualmente la struttura gestionale e informativa e investendo piccoli capitali alla volta;

— Maggiore qualità del prodotto.

La tendenza di questi sistemi verso una crescente complessità e minore ingombro è accompagnata da una sempre maggiore versatilità, come dimostrato dall'aumento nel numero medio di parti prodotte in un FMS. Solo fino a dieci anni fa, i dati contenuti in alcuni studi indicavano un grado di sfruttamento delle potenzialità degli FMS ancora abbastanza ridotto: Lewis (1981) riportava un tempo di utilizzo, nel migliore dei casi, del 60–65%; perdipiù, tale percentuale scendeva al crescere del numero di tipi diversi di prodotti. In effetti, la causa di ciò può vedersi nel fatto che, data la complessità intrinseca di un FMS, il modo in cui vengono allocate le risorse ad un prodotto influenza le modalità di allocazione a tutti gli altri (Suri e Whitney 1982).

Come riportato in (Jaikumar e Van Wassenhove 1989), da un'analisi effettuata su 155 FMS in tutto il mondo, il numero medio di diversi prodotti fabbricati in un FMS è di 80. La percentuale di tempo misurata in cui ogni macchina è in funzione è risultata essere del 92% circa. La riuscita introduzione dei sistemi flessibili ha portato in molti casi a:

— Tempo di operazione (ossia, percentuale di tempo in cui il centro effettivamente lavora) superiore al 75%;

— Produzione di oltre 100 tipi diversi di prodotti all'anno;

— Il *turnover* produttivo (nuovi prodotti che ne rimpiazzano di vecchi) è superiore a 25 tipi di prodotto all'anno;

— Progressiva tendenza verso un funzionamento completamente automatico degli impianti.

— Flessibilità molto elevata, al punto che molti pezzi possono essere lavorati completamente da una sola macchina;

— Basso work in process.

Allo stato attuale, centinaia di FMS sono operativi; il paese–guida per quel che concerne le implementazioni di FMS e FAS è il Giappone, sia come numero di applicazioni che come riuscita produttiva (Kusiak 1985).

1.5. Il concetto di flessibilità

1.5.1 Definizioni e misure di flessibilità

Come abiamo visto (§1.3), i centri di lavorazione flessibili possiedono a bordo macchina un magazzino utensili; tali utensili possono essere applicati al mandrino in tempi estremamente ridotti e dunque il centro può variare rapidamente il tipo di operazione. Dunque, un centro di lavorazione di questo tipo è in grado di effettuare lavorazioni anche di tipo molto diverso, virtualmente senza necessità di set–up tra una lavorazione e l'altra. Questa capacità di riconfigurazione è ciò che rende il sistema *flessibile*, per l'appunto.

In effetti, il concetto di flessibilità può essere analizzato da molte angolazioni diverse, non solo dal punto di vista delle operazioni, ma anche da quello delle variazioni quantitative e qualitative nella domanda, nel processo produttivo, nel mix. Molti sono stati i tentativi di definire in modo più o meno sistematico i vari tipi di flessibilità di un sistema flessibile di lavorazione, e in molti casi sono stati proposti criteri per la misurazione di tali flessibilità. Almeno 50 termini diversi sono stati usati per definire vari tipi di flessibilità, spesso usando terminologie diverse per indicare lo stesso tipo (Sethi e Sethi 1990). Storicamente, comunque, l'uso del termine "flessibile" è stato dapprima riferito agli aspetti economici ed organizzativi dei sistemi produttivi, prima ancora che da quello propriamente operativo.

Dal punto di vista economico, il termine "flessibilità" è stato spesso riferito alla varietà di scelte che si rendono possibili disponendo di un certo capitale, e che può diminuire allorché alcune scelte precludono successivi sviluppi (Marschak e Nelson 1962). Mantenere dunque un adeguato livello di flessibilità sembra dunque essere il miglior modo di premunirsi contro scenari di mercato mutevoli e incerti. Rosenhead et al. (1972) definiscono e misurano la flessibilità come il numero di opzioni alternative che rimangono dopo che viene presa una decisione iniziale. Questo concetto viene messo a fuoco meglio di tutti da Jones e Ostroy (1984), i quali osservano che quanto più mutevoli sono le valutazioni del decisore, tanto più

flessibile sarà la posizione che lui sceglierà (ossia, tanto più ampio sarà il range di opzioni che si vorrà lasciare aperte): "Le posizioni flessibili non sono attraenti in quanto garantiscono un sicuro ritorno di investimento, ma in quanto sono un buon serbatoio di opzioni". Nello stesso lavoro, Jones e Ostroy lamentano che, nonostante la sua fondamentale importanza, il concetto di flessibilità non sia mai stato oggetto di analisi formali in campo economico, forse anche per una difficoltà oggettiva nel trovarne una definizione che non sia dipendente dal particolare modello e/o sistema economico in esame.

Dal punto di vista dell'organizzazione della fabbrica e delle risorse umane, sono state prese in esame varie problematiche legate alla capacità strutturale di un'organizzazione di riconfigurarsi per far fronte a mutate esigenze lavorative. Preece (1986) chiama *flessibilità strutturale* la capacità di un'organizzazione di mettere in grado i propri membri di reagire a cambiamenti nelle condizioni lavorative. Un concetto analogo è quello di *flessibilità della manodopera* (*labor flexibility*) che riguarda la facilità con cui è possibile cambiare i compiti assegnati ad un gruppo di lavoratori per far fronte alle mutate esigenze produttive (Atkinson 1985).

Il concetto di flessibilità rispetto alle operazioni è andato assumendo una rilevanza sempre maggiore dalla fine degli anni '60 in qua. Fino ad allora, alcuni lavori pionieristici si erano limitati a ipotizzare macchine in grado di svolgere "gruppi di operazioni", eventualmente connesse da un sistema automatico di trasporto, ma queste idee non furono sviluppate fino all'avvento dei microprocessori. Da allora, il problema fondamentale è stato quello di riuscire a coniugare la flessibilità propria dei sistemi job–shop con i vantaggi economici della produzione di massa, fino ad allora inscindibilmente legata all'uso di transfer lines o comunque di sistemi produttivi orientati ad un particolare tipo di prodotto. Gli FMS nascono appunto allo scopo di soddisfare questa esigenza, realizzata in misura tanto maggiore quanto più è possibile ridurre i tempi di set–up tra un tipo e un altro di lavorazione e/o di prodotto. Alcuni (Panzar and Willig 1981) hanno inteso indicare tale prerogativa degli FMS col termine *economy of scope* (che potremmo tradurre con *economia di alternative*) contrapposto all'*economia di scala* tipica delle transfer lines.

Anche se il concetto di flessibilità si precisa e si particolarizza in molti modi diversi asseconda del problema e del livello di aggregazione in oggetto, con esso si intende, in generale, quella capacità del sistema di adattarsi a diverse possibili situazioni. Questa capacità di adattamento implica la possibilità di riconfigurare opportunamente le risorse produttive – siano esse macchine, utensili, persone o capitale – per fare fronte a cambiamenti di scenario, sia *interni* che *esterni* all'azienda

(Garrett 1986). Per ciò che concerne la variabilità interna, occorre tenere in conto la possibilità di eventi quali rottura di macchine, variazioni nei tempi di lavorazione, difetti di produzione che costringono ad una parziale rilavorazione (rework), e così via. Fattori esterni all'azienda che spingono per un aumento della flessibilità sono quelli legati, chiaramente, alla variabilità della domanda produttiva, sia in termini qualitativi che quantitativi. In effetti, essendo la domanda di prodotti finiti tipicamente una variabile aleatoria (peraltro spesso di difficile caratterizzazione e stima), l'utilità di disporre di un sistema flessibile appare quasi scontata; tuttavia, anche supponendo di conoscere perfettamente l'evoluzione nel tempo della domanda produttiva, nondimeno la flessibilità del sistema può avere un rilevante impatto strategico (Hayes e Wheelwright 1984, Lim 1987), dal momento che può evitare all'imprenditore l'investimento in nuove risorse, semplicemente riconfigurando o riconvertendo quelle attualmente disponibili.

L'importanza che è stata progressivamente riconosciuta alla flessibilità ha portato molti autori ad uno studio sistematico e approfondito di questo concetto, nonché dei modi più opportuni di definirlo quantitativamente e di misurarlo. Si sono perciò andate affermando terminologie diverse e indipendenti, chiamando talvolta con nomi diversi lo stesso concetto, o, al contrario, usando lo stesso nome per indicare invece diversi tipi di flessibilità. Tra gli studi che hanno tentato di portare un po' di ordine nelle terminologie, vanno menzionati quello di Browne et al. (1984), e quello di Sethi e Sethi (1990). In ambedue i casi, viene proposta una classificazione minuziosa dei vari tipi di flessibilità, spaziando dal livello operazionale (più basso, nella gerarchia decisionale) a quello economico–strategico. Anziché riportare per intero quest'opera di classificazione, ci limiteremo in questa sede a richiamare le definizioni più rilevanti nell'ottica di questo testo, e a proporre noi stessi, nel prossimo §1.5.2, una possibile misura di flessibilità che in qualche modo riassume e include alcune di quelle già note in letteratura. In vari punti del testo, sarà poi sottolineato l'uso più o meno implicito che di tale definizione si è fatto. Va comunque sottolineato che attualmente non esiste una tassonomia universalmente accettata, e che molte di queste definizioni di flessibilità non sono indipendenti l'una dall'altra. Anzi, una critica che si può muovere, forse, a questi tentativi di sistematizzazione, è proprio il fatto di voler talvolta scendere troppo in dettaglio e non cogliere il fatto che tipi diversi di flessibilità hanno un'origine comune — ad esempio, nelle macchine multiutensile.

Tipi di flessibilità *Misure proposte*

Flessibilità di macchina (machine f., equipment f.): Capacità della macchina di eseguire varie operazioni passando dall'una all'altra senza costi o tempi eccessivi.	Numero di operazioni che possono essere compiute (Brill e Mandelbaum 1987); numero di utensili o di part program che la macchina può eseguire (Tarondeau 1982); rapporto tra quantità prodotta e costo di idle in un dato periodo (Son e Park 1987)
Flessibilità di processo (operation f., operation sequence f.): Proprietà del pezzo di essere prodotto per mezzo di processi alternativi	Numero di processi distinti per produrre quel pezzo (Browne et al. 1984, Chatterjee et al. 1987)
Flessibilità rispetto al mix (process f., mix f., job f., part–mix f., variant f.): Capacità del sistema di produrre pezzi di tipi diversi senza interventi costosi sul sistema	Numero di tipi di pezzi distinti in produzione nel sistema (Jaikumar 1986); tempo necessario a variare il mix produttivo di prodotti noti (Warnecke e Steinhilper 1982)
Flessibilità di instradamento (routing f., machine routing f., scheduling f., process f.): Capacità del sistema di produrre lo stesso pezzo attraverso distinti instradamenti, ovvero diverse sequenze di macchine	Numero medio di possibili instradamenti per i vari tipi di pezzi (Chatterjee, Cohen e Maxwell 1987); densità del grafo delle connessioni tra macchine (Carter 1986, Primrose e Leonard 1984)
Flessibilità di volume (volume f., demand f.): Capacità del sistema di produrre a diversi volumi produttivi senza necessità di riconfigurazione delle risorse.	Volume minimo di tutti i tipi di pezzi che è conveniente produrre (Browne et al. 1984); capacità in surplus (slack capacity, Sethi e Sethi 1990)
Flessibilità di mercato (market f.): capacità del sistema di adattarsi ad un ambiente di mercato mutevole	(Inverso del) costo sostenuto per introdurre un nuovo prodotto, aumentare la propria capacità produttiva di una unità, variare il volume di produzione (Sethi e Sethi 1990)

1.5.2. La flessibilità come capacità di riconversione

In questo paragrafo proponiamo anche noi una misura di flessibilità, cercando di essere un po' più "sintetici" rispetto all'insieme di definizioni di flessibilità già presenti in letteratura. Si tratta di una definizione di tipo "aggregato", nel senso che è relativa alla capacità del sistema di riconvertire le proprie risorse tra compiti diversi.

1.5.2.1. *Il grafo bipartito delle operazioni*

Un sistema di produzione viene costruito per far fronte a domande produttive incerte. Globalmente, la domanda produttiva di un certo periodo prefissato (settimana, mese, anno) si realizza in un insieme di ordini di vari tipi di prodotti. Il soddisfacimento di ciascun ordine richiede l'esecuzione di un insieme di operazioni elementari (§1.3.1). Ogni operazione elementare deve far parte di un "repertorio" di operazioni che il sistema è in grado di compiere con almeno una delle sue unità operatrici. L'insieme di operazioni che il sistema è in grado di eseguire è l'*insieme base* di operazioni. Sia N tale insieme, e n la sua cardinalità.

Asseconda del sottoinsieme di operazioni–base che una macchina può eseguire, parleremo di *tipi di macchine*; ossia, due macchine saranno per noi dello stesso tipo se sono in grado di eseguire lo stesso insieme di operazioni base. Sia M ($|M|=m$) l'insieme dei tipi di macchine da utilizzarsi per la produzione.

Sia P un m-vettore intero, nonnegativo, tale che p_j è pari al numero di macchine di tipo j presenti nel sistema, mentre $p=\Sigma_j p_j$ è il numero totale di macchine.

Si consideri l'insieme di prodotti richiesti dal mercato in un certo periodo. Per ogni unità di ogni tipo di prodotto, si può pensare di conoscere il tempo richiesto per ciascuna delle operazioni base del sistema. Moltiplicando tali quantità per il numero di unità da produrre di quel tipo, si ottiene un n–vettore d (*piano di produzione*) tale che d_i è pari al tempo di operazione i complessivamente richiesto per soddisfare quella domanda produttiva.

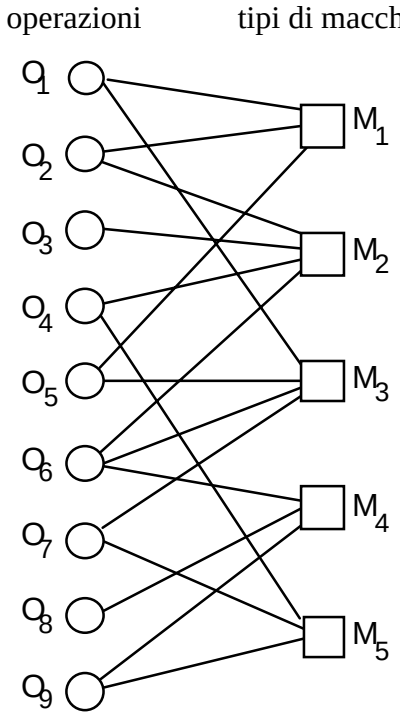


Figura 1.4. Macchine e operazioni eseguibili.

Ogni unità operatrice del sistema può strutturalmente eseguire un certo sottoinsieme delle operazioni base. Infatti, un tornio a controllo numerico potrà effettuare diversi tipi di operazioni di taglio rotazionale, ma non, ad esempio, un'operazione di perforatura. Possiamo allora rappresentare la situazione per mezzo di un grafo bipartito $G(N,M,A)$ in cui (vedi Fig.10.4):

— ogni nodo sulla sinistra corrisponde a un tipo di operazione;

— ogni nodo sulla destra corrisponde a un tipo di macchina;

— c'è un arco da $i \in N$ a $j \in M$ se e solo se una macchina di tipo j può eseguire l'operazione i.

Questo grafo è concettualmente analogo a quelli che verranno introdotti nei §§4.3 e 4.4. Difatti, il problema di attrezzaggio descritto nel §4.3 fa implicitamente riferimento al concetto di flessibilità che stiamo per introdurre.

Sia $X(P)$ l'insieme di punti di $\leftarrow^n$ che rappresentano le domande produttive soddisfacibili da un certo sistema produttivo. L'insieme $X(P)$ è, per ogni P, un politopo ottenuto come combinazione convessa di:

— un insieme p di punti di $\leftarrow^n$ a componenti non negative;

— l'insieme di punti di $\leftarrow^n$ otenuti dai punti in p sostituendo, in tutti i modi possibili, una o più componenti con 0.

Dunque, tutti e soli i punti di *X(P)* corrispondono a domande produttive soddisfacibili nell'intervallo di tempo preso in considerazione. Peraltro, si consideri un insieme *D* di possibili domande produttive; e supponiamo di sapere che, in ogni periodo di ineteresse, *una e una sola* di esse si avvererà. Possiamo rappresentare anche i punti di *D* in $\leftarrow^n$. Sia *?* l'insieme dei vertici della combinazione convessa di *D*. Chiaramente, un sistema produttivo può soddisfare qualunque punto dell'insieme *D* se e solo se risulta *X(P) ?*.

1.5.2.2 *Misura della flessibilità*

La flessibilità di un sistema di produzione è per noi la *capacità del sistema di riallocare le proprie risorse a fronte di variazioni nella domanda, nel rispetto dei vincoli operativi delle unità operatrici e senza far diminuire il livello produttivo*.

Di seguito, vediamo quale forma assume *X(P)* per il semplice modello di sistema che stiamo considerando. questo ci condurrà ad una definizione di flessibilità.

Supponiamo che il nostro sistema debba eseguire solo due tipi diversi di operazioni (n=2). Le Figure 1.5(a)-(b) rappresentano due possibili situazioni di domande che possono essere soddisfatte in un periodo di lunghezza T. Nel seguito, sia p_1 il numero di macchine in grado di eseguire l'operazione O_1, p_2 il numero di macchine in grado di eseguire l'operazione O_2, e p_{12} il numero di macchine in grado di eseguirle entrambe ($p_1+p_2+p_{12}=p$).

Nel caso (a), il sistema può destinare l'intero budget di tempo pT in cui le macchine sono disponibili totalmente all'operazione 1 o totalmente all'operazione 2. Di conseguenza, qualunque composizione relativa di queste due domande – tale che la loro somma non ecceda pT – può essere realizzata. In questo caso il sistema è completamente flessibile, dal momento che durante un periodo di tempo di lunghezza T può allocare la propria capacità in modo da soddisfare qualunque domanda non superiore a pT.

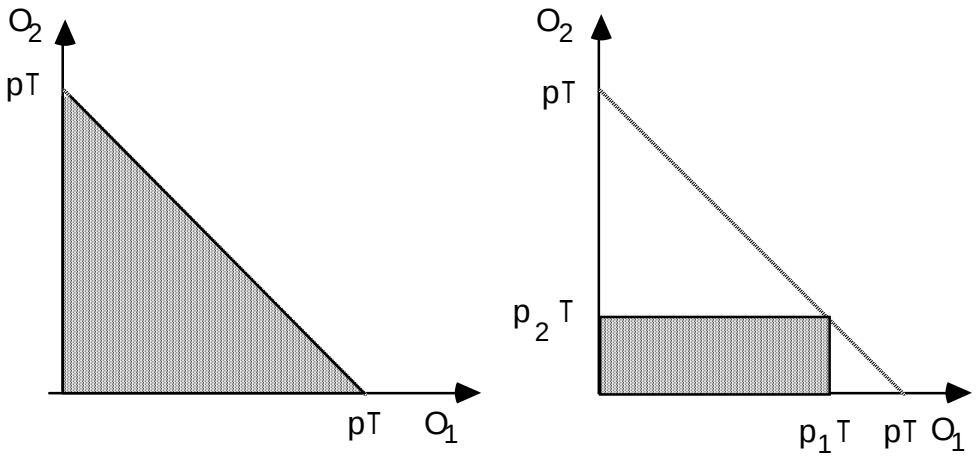


Figura 1.5. Due casi estremi per *X(P)* in un sistema con due operazioni:
(*a*) sistema totalmente flessibile (*b*) sistema rigido

Nel caso (b), il sistema può eseguire l'operazione O_1 fino ad una quantità p_1T, e O_2 fino a p_2T: chiaramente, in questo caso il sistema consiste di due soli diversi tipi di macchine, ognuna in grado di eseguire solo uno dei due tipi di operazione. Questo tipo di soluzione è dunque *rigido* e non è possibile elargire i due tipi di operazioni in rapporto diverso da p_1/p_2, senza rinunciare ad avere un pieno utilizzo del sistema.

Il caso più generale è illustrato in Fig.1.6: il più grande ammontare di operazione O_1 che può essere eseguito nel periodo T vale $(p_1+p_{12})T$; se il tempo p_2T è allocato all'operazione O_2 otteniamo la piena utilizzazione del sistema (punto B). Similmente, un altro punto ammissibile può trovarsi cercando di allocare quanta più capacità possibile all'operazione O_2 (punto A). Quindi, *X(P)* è completamente nota.

A parte un fattore v2, la lunghezza del segmento AB rappresenta la porzione dell'intero carico di lavoro che il sistema è chiamato a svolgere che può essere liberamente allocata tra le due operazioni, mantenendo l'utilizzazione del sistema al suo massimo valore teorico. Quindi, sembra ragionevole prendere questa lunghezza come *misura della flessibilità del sistema*. Si noti che questo valore è pari al numero di macchine in grado di eseguire entrambe le operazioni O_1 and O_2 – moltiplicato T. Agnetis, Lucertini e Nicolò (1990) propongono un modo di estendere questa definizione al caso $n>2$, sempre come funzione della composizione del sistema, ovvero dei valori p_j.

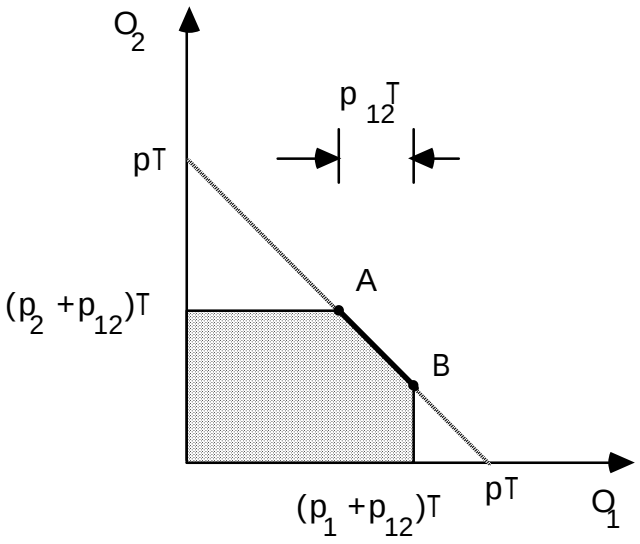


Figura 1.6. Forma generale di *X(P)* in un sistema in grado di svolgere due operazioni.

1.6 Problemi decisionali in FMS e FAS

Nella prima metà degli anni '80, con lo sviluppo delle tecnologie dei sistemi di lavorazione flessibili, si è cominciato a mettere a fuoco anche i problemi decisionali ad essi connessi. Una suddivisione abbastanza comune è quella in problemi di *progetto*, di *pianificazione* e di *scheduling*. Questo testo tratterà solo degli ultimi due tipi di problemi. Nel seguito, utilizzeremo una classificazione simile a quella presentata da Stecke (1985), ma con alcune varianti. I problemi di pianificazione riguardano la gestione dei flussi di materiali all'interno dell'impianto, e si riferiscono a quelle decisioni che vanno prese *prima* che la produzione abbia fisicamente luogo. Le decisioni da prendere consistono nell'allocazione fisica di risorse — utensili, pezzi, intervalli di tempo — e di operazioni alle unità operatrici. L'orizzonte temporale di questi problemi va, tipicamente, da poche ore ad alcuni giorni. Una volta effettuate queste allocazioni, il sistema può iniziare la produzione e a questo punto si pongono problemi relativi al raggiungimento degli obiettivi (ad esempio, in termini di utilizzazione delle macchine) in base ai quali è stata effettuata l'allocazione di risorse. Questi ultimi sono i problemi di *scheduling*, che si pongono in tempo reale.

1.6.1. Problemi di pianificazione

— *Selezione dei tipi di pezzi da produrre* (*part type selection*). All'inizio di un intervallo di produzione (ad es. un giorno), vanno specificati i part type da produrre, tra quelli la cui domanda riguarda il periodo di tempo in considerazione (ad es. durante la settimana attuale). La decisione può avvenire in base alle date di consegna dei pezzi, o in modo da garantire un vantaggioso utilizzo dei

componenti del sistema, come nel modello presentato nel §3.4. Una volta scelti i tipi di pezzi da produrre, rimangono definiti il numero complessivo di utensili che andranno impiegati nel prossimo periodo, e i vari tipi di operazioni;

— *Raggruppamento delle macchine* (*Machine grouping*). Attrezzare diverse macchine con lo stesso insieme di utensili – rendendole così funzionalmente identiche – può essere utile al fine di massimizzare la produttività del sistema.

— *Determinazione dei rapporti di produzione* (*Production ratios determination*). Strettamente connesso al problema di selezione dei part type, consiste nel definire le quantità relative da produrre dei vari tipi di pezzi selezionati: ad esempio, i pezzi A e B devono essere prodotti nel rapporto di 1:3. L'obiettivo è ancora quello, in generale, di massimizzare l'utilizzo delle risorse del sistema. Ovviamente tale problema si pone qualora non esistano particolari vincoli sul valore di questi rapporti di produzione: viceversa, in una situazione in cui i pezzi A e B prodotti vengono assiemati in una linea a valle del sistema in esame, potrebbe risultare necessario produrre A e B nelle quantità richieste dall'operazione di assiematura, in quanto altrimenti quest'ultima starebbe ferma o, al contrario, il buffer potrebbe essere eccessivamente caricato.

— *Allocazione degli utensili* (*tooling*). Sulla base dei pezzi da produrre, va assegnato ad ogni macchina un insieme di utensili, ossia va determinata la configurazione del magazzino utensili di ciascuna macchina. Eventualmente, utensili identici possono essere caricati su macchine diverse, se ciò può risultare vantaggioso. L'obiettivo del tooling è legato alla "qualità" delle soluzioni che possono ottenersi per i successivi problemi di routing e scheduling. Un attrezzaggio potrà quindi essere valutato sulla base di una stima del numero di part transfer cui può dare luogo, o di una stima del bilanciamento dei carichi di lavoro ottenibile con tale attrezzaggio (§4.3). Obiettivi e metodologie per tale problema variano in realtà parecchio asseconda del contesto: in una cella, è possibile che gli utensili siano assegnati dinamicamente a una macchina, e allora il problema consiste nel sincronizzare opportunamente la movimentazione degli utensili (§2.5); in un FMS, l'attività di cambio utensili può avvenire in modo continuativo nel tempo, anziché tra due intervalli di produzione, se viene adottato il cosiddetto *approccio flessibile* al problema di Part type Selection (§3.4.2).

— *Assegnamento delle operazioni alle macchine* (*Routing* o *Loading*). Una volta definito l'insieme di pezzi da produrre e la configurazione dei magazzini utensili di ciascuna macchina, occorre decidere, per ogni pezzo che dovrà essere prodotto dal sistema, su quale macchina dovrà essere effettuata ciascuna delle operazioni

richieste. Tale assegnamento definisce l'*instradamento* di ciascun pezzo. L'insieme degli instradamenti di tutti i pezzi in produzione, definisce un *routing*. Il problema consiste quindi nel trovare il routing più conveniente. Gli obiettivi di tale problema riguardano tipicamente la massimizzazione della produttività, ossia, in definitiva, dell'utilizzo delle macchine. In un FMS, tale obiettivo si traduce tipicamente nella necessità di bilanciare i carichi di lavoro tra le macchine, tenendo però conto della necessità di non sovraccaricare il sistema di trasporto, e dunque un obiettivo è anche quello di minimizzare i costi complessivi di trasferimento dei pezzi da una macchina all'altra (§4.4). Per architetture pipeline, è possibile affrontare l'obiettivo della minimizzazione del tempo di completamento di un insieme di pezzi (§3.2). In una cella per produzione omogenea, inoltre, l'obiettivo può porsi come massimizzazione del throughput, ossia del numero di pezzi prodotti nell'unità di tempo (§2.4). In letteratura, spesso viene affrontato un problema diverso, indicato col termine di *loading*: assumendo che esista una sola copia di ogni utensile, il problema è quello di assegnare gli utensili alle macchine e dunque, di fatto, anche le operazioni: in altre parole, i due problemi di tooling e routing vengono risolti allo stesso tempo. In letteratura, anche se esistono formulazioni che prevedono la possibilità di duplicare utensili su macchine diverse (§4.2), non viene mai considerato l'insieme degli instradamenti corrispondenti a un certo attrezzaggio, come viene fatto invece nel modello sviluppato nel §4.4.

1.6.2. Problemi di scheduling

I problemi di scheduling per FMS riguardano la determinazione della sequenza in cui è più conveniente introdurre i pezzi nel sistema e in cui processare i pezzi su ciascuna singola macchina.

Una volta risolto il problema di routing, gli instradamenti di tutti i pezzi sono fissati. Il problema è allora quello di ricercare una schedulazione di dettaglio per ciascun pezzo che entra nel sistema, con l'obiettivo, ad esempio, di minimizzare il tempo di completamento di tutti i pezzi. Spesso tale problema ricade nel formato di un problema di JOB SHOP (French 1982), o di problemi di scheduling più particolari, qualora la struttura del sistema sia meno generale: ad esempio, nel §3.3 vedremo un problema di sequenziamento di pezzi in un sistema pipeline privo di buffer.

Talora, tuttavia, il numero di centri di lavorazione e quello, in genere elevato, di pezzi da lavorare, rende pressoché impossibile trattare tale problema attraverso algoritmi di scheduling, per quanto efficienti, e si fa ricorso invece a regole euristiche da applicarsi *in tempo reale*: ad esempio, nel momento in cui un centro di lavorazione

si libera, il successivo pezzo da caricare sul centro può essere prescelto attraverso una regola del tipo "seleziona il pezzo per cui il tempo residuo è massimo" o simili. La decisione circa il prossimo pezzo da introdurre nel sistema può basarsi, invece, sulla volontà di preservare un certo mix produttivo tra i part type in corso di produzione.

L'insieme di possibili regole decisionali per problemi di scheduling in FMS è estremamente ampio; diversi lavori sono stati svolti per valutare comparativamente l'efficacia delle diverse regole, attraverso dei modelli di simulazione. Tra i contributi più significativi, menzioniamo quelli di Panwalker e Iskander (1977) e di Gupta, Gupta e Bector (1989).

1.7 Modellistica per sistemi flessibili

1.7.1 Approcci modellistici per FMS e FAS

I sistemi flessibili di lavorazione (FMS) sono per loro natura caratterizzati, dal punto di vista operativo, da un insieme di parametri il cui numero può essere estremamente elevato: basti pensare alle diverse tipologie di utensili, centri di lavorazione, pezzi che il sistema può essere chiamato rispettivamente a gestire e produrre; alle strutture, talora molto diverse, del sistema di trasporto. Come visto nel §1.3, un FMS contiene un insieme di risorse estremamente vario e numeroso. Corrispondentemente, i problemi decisionali (§1.6) connessi con la gestione di queste possono risultare estremamente complessi, anche a causa del forte grado di interazione esistente tra le diverse decisioni, per cui, talora, anche semplicemente trovare il modo più significativo per decomporre il problema complessivo in diversi problemi trattabili è un compito non banale.

La decisione di investire in sistemi flessibili di produzione richiede dei criteri di valutazione nuovi rispetto a quelli "classici" (es. tasso di rendimento interno), e ciò tra l'altro può costituire un fattore frenante nei confronti di investimenti innovativi: infatti, i costi della flessibilità devono essere valutati principalmente nel quadro della pianificazione a medio-lungo termine sulla base di stime della diversificazione produttiva. D'altra parte è difficile quantificare la flessibilità del sistema in modo confrontabile con altri parametri di valutazione. Risulta allora utile disporre di strumenti rappresentativi che consentano da un lato di ottenere indicazioni sulle scelte operative (configurazione del sistema, algoritmi di allocazione, routing, sequenziamento etc.); e dall'altro di verificare le scelte mediante simulazione del funzionamento del sistema per la valutazione sia del trade-off produttività/costi sia delle caratteristiche di flessibilità.

Strumenti di questo tipo devono evidentemente ipotizzare una qualche rappresentazione del sistema: è necessario quindi passare attraverso una fase di *modellistica*. D'altro canto, in uno scenario complesso come quello degli FMS non è semplice concepire un modello che abbia tutti i requisiti che l'utente dello strumento richiede, e che sono spesso in conflitto tra loro. Infatti, tipicamente, quanto più un modello cerca di rappresentare in dettaglio il sistema reale, e dunque di tener conto di tutte le sue caratteristiche nel processo di ottimizzazione, tanto più i suoi risultati saranno significativi, ma ciò a prezzo di un aumento del carico computazionale e, principalmente, della complessità d'uso dello strumento che ne deriva.

In ogni caso, qualunque modello sarà sempre una "semplificazione della realtà" (Buzacott 1987): perché possa essere utile in pratica, un modello potrà includere solo alcune delle caratteristiche del sistema e, in particolare, dovrà concentrarsi su quelle che influiscono sugli indici di prestazioni di interesse, al livello di aggregazione desiderato. Ad esempio, nella pianificazione della produzione di medio–lungo termine (un problema che non trattiamo in questo testo) è chiaramente inutile avere informazioni di dettaglio su utensili, tempi e successione di operazioni richieste da ogni singolo pezzo che il sistema dovrà produrre, mentre viceversa questo tipo di informazioni sono richieste nella pianificazione operativa e nei problemi di scheduling.

In questo capitolo, vogliamo brevemente discutere i tre principali approcci modellistici impiegati nello studio degli FMS, che sono: la simulazione, l'analisi e l'ottimizzazione.

1.7.2 Modelli di simulazione

Tra le classi di modelli che hanno ricevuto notevole attenzione nella letteratura degli FMS, i *modelli di simulazione* hanno un ruolo di primo piano (Law e Kelton 1982). La simulazione è uno strumento modellistico che consente, in pratica, di associare oggetti del mondo reale, come pezzi, utensili, pallet, a byte: spostando questi byte da una locazione all'altra di memoria, il computer simula lo spostamento dei pezzi da un centro di lavorazione ad un altro. Dunque, attraverso una rappresentazione dettagliata del sistema e delle politiche di controllo prescelte, il modello di simulazione riproduce il comportamento del sistema stesso, e consente di osservare l'andamento dei valori dei parametri ritenuti significativi. La struttura dei modelli di simulazione è tipicamente "modulare", nel senso che è in genere possibile rappresentare un sistema più grande (ossia che ingloba il precedente), senza dover modificare radicalmente la struttura del modello stesso. In definitiva, i modelli di simulazione appaiono uno

strumento abbastanza naturale per effettuare una *verifica* di scelte operative effettuata sulla base di un modello decisionale, di cui si parlerà in seguito (§1.7.4).

Esistono varie classi di modelli di simulazione. La più diffusa è forse quella dei modelli "a rete", in cui vengono definiti dei simboli grafici corrispondenti a entità materiali (es. macchine), di cui va specificata la relativa posizione fisica (distanze). L'esistenza, oggi, di linguaggi di simulazione potenti (SIMAN, SLAM, solo per citarne due) e di interfacce grafiche, ha enormemente facilitato il compito di costruire un modello di simulazione e di osservarne l'esecuzione, allargando di molto la schiera dei potenziali utilizzatori di modelli di simulazione.

Un'osservazione di carattere generale riguardante i modelli di simulazione è che, per essere definito e per girare, un modello di simulazione ha bisogno della specificazione di una grande quantità di dati: i tempi operazione di ogni singolo pezzo, il mix, la dislocazione fisica delle macchine, il tempo necessario per andare da ogni macchina a ogni altra, la politica di scheduling dei pezzi, gli instradamenti dei singoli pezzi. Ottenere queste informazioni – relative, ad esempio, ad un impianto già in funzione – può essere un compito lungo e non semplice, e può comportare costi considerevoli. Inoltre, l'esecuzione di modelli di simulazione richiede in genere tempo e risorse di calcolo non trascurabili. D'altro canto, non sempre sono richieste valutazioni di dettaglio come quelle fornite dai modelli di simulazione; una possibile alternativa sono allora i modelli analitici.

1.7.3. Modelli analitici

L'impiego di un modello di simulazione di un FMS presuppone che le caratteristiche del sistema siano note ad un considerevole livello di dettaglio. D'altro canto, dal punto di vista di chi sta compiendo uno studio di fattibilità per il dimensionamento e l'acquisto di un FMS, non tutti i dettagli del sistema saranno noti o prevedibili: infatti, perfino l'insieme di pezzi che il sistema potrà essere chiamato a produrre può non essere noto con esattezza. Di conseguenza, la struttura dell'FMS potrebbe essere una qualsiasi di un insieme di soluzioni, ugualmente soddisfacenti; il problema è perciò quello di specificare in modo sempre più preciso il progetto di un FMS, attraverso un procedimento consistente in una valutazione iterativa in cui, ad ogni passo, occorre valutare la "bontà" di una possibile soluzione. Uno strumento che consenta analisi "di massima" ma attendibili circa la qualità di una possibile configurazione del sistema può far risparmiare tempo e risorse di calcolo, consentendo poi di concentrarsi e quindi di condurre analisi di dettaglio – con modelli di simulazione – solo su quelle configurazioni che hanno pasato indenni questo primo "setaccio". Tale strumento deve essere volto a stimare misure delle prestazioni del sistema simili a quelle stesse

già espresse dai modelli di simulazione, ma facendo uso di una quantità molto più ristretta di informazioni. La classe di modelli che ha i requisiti richiesti è quella dei *modelli analitici*.

I modelli analitici per FMS sono stati prevalentemente sviluppati, verso la fine degli anni '70, impiegando le metodologie delle *reti di code*, sia *aperte* (Buzacott e Shantikumar 1980) che *chiuse* (Solberg 1977). L'intero FMS può essere rappresentato come una rete di stazioni di servizio, ciascuna caratterizzata da tempi di servizio aventi distribuzione esponenziale, e in cui gli arrivi di pezzi dall'esterno (se la rete è aperta) obbediscono a una distribuzione poissoniana. Si suppone che anche gli instradamenti dei pezzi siano noti probabilisticamente, in particolare si suppone di conoscere la probabilità che il generico pezzo in uscita da una macchina ha di essere instradato ad altre macchine. Con le metodologie delle reti di code, è possibile ottenere informazioni su alcune grandezze di estremo interesse pratico, quali ad esempio il throughput (numero di pezzi prodotti nell'unità di tempo), il tempo medio trascorso nel sistema dal generico pezzo, il valor medio del work in process. Si noti che l'assunzione che la rete di code sia chiusa corrisponde alla circostanza, spesso verificata in pratica, che il numero di pallet presenti nel sistema sia limitato, e che sono costantemente tutti occupati (non appena un pezzo esce dal sistema esso viene rimpiazzato da uno nuovo, che viene montato sullo stesso pallet del pezzo appena uscito).

Rispetto ai modelli di simulazione, si può osservare che i modelli analitici sono più trasparenti e flessibili rispetto a variazioni nei valori dei parametri, mentre lo sono di meno rispetto a variazioni strutturali del modello: riuscire a esprimere la relazione tra due grandezze (ad esempio, il throughput in funzione del work in process) in forma chiusa è certamente un risultato estremamente interessante e utile: in generale, diagrammare la stessa relazione facendo uso di un modello di simulazione richiede l'esecuzione ripetuta dello stesso modello e l'interpolazione di punti (anche se oggi esistono tecniche per limitare il numero di simulazioni richieste, come ad esempio la Perturbation Analysis (Ho 1987)).

Uno dei punti deboli dell'approccio analitico sta nel fatto che, al fine di avere modelli computazionalmente efficienti, è necessario fare ipotesi che si potrebbero rivelare talvolta non realistiche: ad esempio, molti modelli analitici suppongono che i tempi di servizio siano esponenziali, quando invece, in un FMS (e in genere in tutti i sistemi automatizzati) essi sono sensibilmente deterministici. C'è comunque da sottolineare la relativa robustezza di molti risultati cui la teoria delle code perviene

(Suri 1983), ed i numerosi tentativi di rimuovere questa limitazione (ad esempio quelli di Yao e Buzacott (1985)).

1.7.4. Modelli di ottimizzazione

Come detto nel §1.7.1, l'utente di FMS vuole effettuare le sue scelte operative al fine di conseguire certi obiettivi, che in genere riguardano la minimizzazione dei costi (a fronte di una domanda di prodotto finito da soddisfare), la massimizzazione della produttività (data una certa disponibilità di risorse), la flessibilità, o, più realisticamente, un compromesso tra questi ed altri obiettivi. E' necessario cioè pensare all'esistenza di un livello decisionale il cui ruolo è di fornire le decisioni operative più "convenienti" rispetto ai suddetti obiettivi. A questo livello decisionale occorre avere una descrizione degli *obiettivi* inerenti configurazione e gestione dell'FMS: si rende necessario l'uso di un *modello di ottimizzazione*, concepito come *strumento di rappresentazione del sistema orientata alla soluzione dei problemi*.

Rispetto a un modello di simulazione o a un modello analitico, un modello di ottimizzazione dà una descrizione diversa del sistema, caratterizzata dal fatto di essere orientata a generare delle "regole di comportamento" (ad esempio, l'ordine in cui eseguire certe lavorazioni). La costruzione di un modello di ottimizzazione costringe ad effettuare un'analisi il più possibile attenta ed approfondita dei meccanismi con cui la scelta e l'allocazione delle risorse influenza quelle grandezze ritenute significative dall'utente del modello: attraverso la comprensione di tali meccanismi sarà possibile "guidare" scelte e allocazioni in modo ottimo.

Come accennato in precedenza, le dimensioni del problema di ottimizzazione complessivo possono facilmente essere fuori della portata dei mezzi di calcolo disponibili (soprattutto se si suppone che l'utente del modello non disponga di calcolatori molto potenti). In tal caso, il modello di ottimizzazione dovrà allora *decomporre* il problema decisionale complessivo in un insieme (strutturato) di sottoproblemi trattabili, tali che la composizione delle loro soluzioni ottime fornisca una buona soluzione – ancorché non necessariamente ottima – del problema originario. Scopo di questa decomposizione è quello di riconoscere ed isolare le parti "*facili*" – ossia sottoproblemi che possono essere risolti efficientemente in modo esatto – da quelle "*difficili*", ossia per le quali l'unico approccio praticabile sembra essere quello di una procedura euristica.

Modelli di simulazione e analitici da un lato, e di ottimizzazione dall'altro, hanno pregi e difetti in un certo senso complementari. Infatti, i primi sono concepiti per avere informazioni affidabili sul comportamento reale del sistema, e richiedono tempi

di elaborazione – nel caso dei modelli di simulazione – solitamente consistenti; mentre ai secondi è richiesto di generare, in tempo ragionevole, delle scelte di configurazione e politiche di controllo coerenti con gli obiettivi operativi. Dal punto di vista concettuale, dunque, si può pensare di integrare i due approcci in un unico strumento che, attraverso la simulazione del modo "ottimo" – secondo un certo modello di ottimizzazione – di operare del sistema, fornisca informazioni sia sulla "bontà" del comportamento effettivo del sistema, sia su come, dal punto di vista operativo, si possa far funzionare il sistema in tali condizioni, ovvero su quali parametri conviene agire per migliorare la soluzione trovata.

In questo testo verranno presentati dei modelli di *ottimizzazione combinatoria*, ossia in cui il problema è formulabile attraverso un insieme di variabili decisionali, ciascuna delle quali può assumere una quantità finita di valori (spesso sono solo due: 0 o 1).

I modelli sviluppati sono *deterministici*, il che appare giustificato in uno scenario in cui tutte le operazioni sono automatizzate e dunque i tempi di esecuzione delle varie attività (lavorazioni, scambio utensili, trasferimento delle navette etc.) sono noti con ragionevole certezza, mentre l'intervento umano è ristretto a quelle sole situazioni in cui hanno luogo eventi "esterni" al normale funzionamento del sistema: guasto di una macchina, improvvisa indisponibilità di un utensile per fine vita o rottura etc. — situazioni che, generalmente, possono essere considerate come altrettanti momenti di ricalcolo dell'"ottimo".

In definitiva, i modelli e gli algoritmi che verranno illustrati in questo testo possono essere pensati come i componenti fondamentali di un *sistema integrato di supporto alle decisioni* (*SISD*). Lo scopo di tale sistema è quello di essere utilizzabile da un cliente che, volendo soddisfare determinate specifiche riguardanti principalmente costi e produttività, si chiede quale *configurazione* debba avere il sistema al fine di conseguire queste specifiche nell'ipotesi che esso venga gestito "al meglio" secondo i modelli di ottimizzazione inglobati nel *SISD*. L'input al *SISD* viene dato quindi dal *cliente*, in termini sia di richieste che vuole siano soddisfatte, sia delle scelte di configurazione e di gestione che non sono affidate ai modelli di ottimizzazione. Il *SISD* deve dunque tener conto delle varie architetture/configurazioni possibili che il *sistema* produttivo può assumere.

1.8 Classificazione e nomenclatura di FMS e FAS

Jaikumar e Van Wassenhove (1989) propongono una classificazione, dal punto di vista strutturale, dei sistemi flessibili di produzione oggi esistenti nel mondo. Non è

chiaramente l'unico tipo di classificazione che è stato tentato (interessante ad esempio anche quella presentata da Rachamadugu e Stecke (1987)); tuttavia, vale la pena di soffermarvisi in quanto, partendo dall'analisi effettuata su un campione reale di FMS, consente di introdurre le differenze strutturali fondamentali che ritroveremo tra le diverse classi di modelli sviluppati in questo testo.

La classificazione avviene in base al tipo di interazione esistente tra le risorse del sistema e alle possibili modalità di *instradamento*, ossia di assegnamento delle operazioni alle macchine operatrici. Nel seguito, il termine *risorse* va inteso nel senso di risorse del sistema che sono richieste all'unità operatrice per effettuare le operazioni, escludendo gli utensili. Utilizzando la terminologia che adotteremo nel resto del testo, possiamo distinguere:

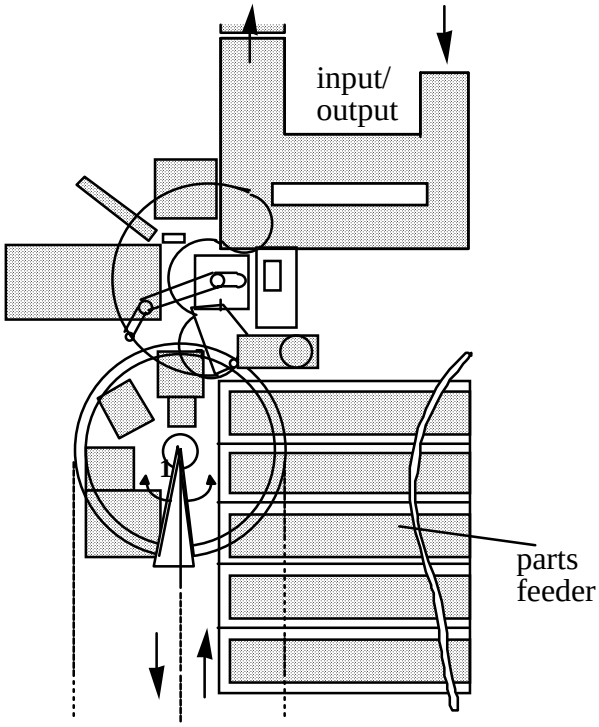


Figura 1.7. Una cella flessibile di lavorazione.

— *Celle flessibili* (Fig.1.7). Sono caratterizzate da un elevato grado di accoppiamento e interazione tra le varie unità operatrici, a causa della condivisione di risorse. Più precisamente, una volta che un pezzo è rilasciato per un'operazione, l'unità operatrice è libera, ed è dotata dell'utensile necessario, l'operazione può avvenire solo se tutte le risorse richieste da quell'operazione non sono attualmente usate da altre unità operatrici; altrimenti, unità operatrice e pezzo devono attendere il rilascio di tutte queste risorse. Risorse di questo tipo sono ad esempio dispositivi di immagazzinamento temporaneo, piani di lavoro, macchine per il testing, attrezzi o utensili speciali; tipicamente, infatti, tali risorse sono presenti in quantità limitata e

dunque è necessario disciplinarne l'accesso. Inoltre, per motivi di semplicità gestionale e di limitato spazio fisico disponibile, il modo di operare più comune è quello in cui *pezzi dello stesso tipo seguono lo stesso instradamento*, vale a dire, come sarà chiarito meglio nel seguito (§2.1), che una stessa operazione viene svolta dalla stessa unità operatrice su tutte le unità. Le unità operatrici possono o meno essere dotate di un sistema locale di immagazzinamento utensili; la presenza o meno di tale magazzino influenza fortemente la natura dei problemi decisionali (§§2.4 e 2.5).

— *Sistemi pipeline o transfer lines* (Fig.1.8). Sono sistemi in cui la topologia del sistema di trasporto è quella di una catena. Tutti i pezzi, anche di tipo diverso, visitano le macchine nello stesso ordine e, in genere, senza mai visitare due volte una stessa macchina; di conseguenza esiste un ordinamento totale tra le macchine. Sono l'evoluzione delle classiche linee di assiematura, in cui le stazioni di lavoro sono costituite da squadre di operai, e sono anche note col nome di *linee di flusso flessibili* (*flexible flow lines*).

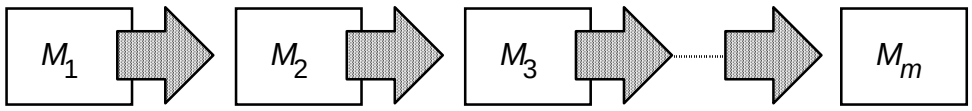


Figura 1.8. Schema di un sistema di produzione pipeline.

— *Stazioni di lavoro indipendenti*, ovvero *Sistemi flessibili di lavorazione* (*FMS*) o *di assiematura* (*FAS*) propriamente detti (Fig.1.2). Tali sistemi sono costituiti da diversi centri di lavorazione indipendenti (*stand–alone*), ognuno dotato di un proprio dispositivo di immagazzinamento sia di parti che di utensili e in cui l'accoppiamento tra diversi centri avviene tramite il solo sistema di trasporto (di parti, ed eventualmente di utensili).

A queste tre diverse classi di sistemi corrispondono problemi decisionali e quindi modelli strutturalmente diversi. Il resto del testo è strutturato secondo questa classificazione. Per quanto concerne le celle flessibili (§2), vedremo il problema della gestione delle operazioni in due scenari, caratterizzati rispettivamente dall'allocazione statica degli utensili alle unità operatrici (§2.3) o dalla gestione dinamica (§2.4). Nel primo caso, oltre a disciplinare l'accesso alle risorse del sistema da parte delle unità operatrici, le decisioni comprendono anche l'assegnamento degli utensili alle unità operatrici; nel secondo caso tale aspetto non è presente, ma il problema di sincronizzazione è in generale più complicato dalla necessità di tenere in considerazione i tempi di spostamento degli utensili nella cella.

I sistemi pipeline sono analizzati nel §3. Vedremo dapprima il problema del routing per una particolare struttura degli alberi di assiematura (§3.2); quindi analizzeremo due problemi che si pongono quando i buffer hanno capacità limitata: nel caso che essi non ci siano (§3.3) proporremo un algoritmo approssimato per il problema dello scheduling dei pezzi; nel caso di pipeline costituito da due macchine e buffer di capacità limitata (§3.4), vedremo un approccio al problema complessivo della selezione dei part type e del sequenziamento dei pezzi.

Infine, vedremo modelli per attrezzaggio e instradamento in FAS (§4). I modelli presentati nei §§4.3 e 4.4 concettualmente possono essere pensati in cascata, in quanto i primi forniscono degli attrezzaggi tali da massimizzare certe funzioni obiettivo, e il secondo calcola degli instradamenti in modo da massimizzare la produttività.

2. ROUTING NELLE CELLE FLESSIBILI DI LAVORAZIONE: ALLOCAZIONE DEGLI UTENSILI E SINCRONIZZAZIONE DELLE OPERAZIONI

2.1 Introduzione

In questo capitolo verranno discussi i problemi di assegnamento delle operazioni nell'ambito di una *cella flessibile di lavorazione* (*FMC*). Come si è già detto nel §1.8, l'ambiente–cella è caratterizzato dal fatto che esiste un notevole grado di interazione tra le risorse produttive, e questo pone tipicamente problemi di *sincronizzazione* che vanno risolti se si vuole massimizzare l'efficienza del sistema. Tale sincronizzazione riguarderà risorse che possono essere le più svariate: superfici di lavoro che vengono utilizzate alternativamente dai diversi pezzi nel sistema, macchinari speciali (es. di test), navette o veicoli per il trasporto di pezzi o utensili etc. Appare allora di estrema importanza, dal punto di vista modellistico, distinguere tra due tipi di risorse, in base al modo in cui le unità operatrici del sistema possono impiegarle. Alcune risorse (tipicamente, utensili) vengono allocate in modo *statico* a certe unità operatrici, e non vengono smontate dalle rispettive unità operatrici prima di un nuovo intervallo produttivo; altre, invece, possono essere scambiate tra le unità operatrici stesse durante il funzionamento del sistema. In effetti, il problema di sincronizzazione si pone per queste ultime, ma la qualità della soluzione del problema complessivo di massimizzare la produttività della cella dipende anche dall'allocazione delle risorse "statiche". Perciò, in generale occorre risolvere due problemi che sono interallacciati (§2.4.2). Se invece tutte le risorse richieste dalle singole operazioni possono essere dinamicamente assegnate alle unità operatrici di volta in volta, il problema di assegnare risorse "statiche" non si pone, mentre rimane quello di sincronizzare gli spostamenti delle risorse e delle strutture di trasporto impiegate per la loro movimentazione.

Dunque, talune coppie di operazioni, anche se vengono assegnate a macchine diverse, non possono essere eseguite in parallelo, se l'insieme delle risorse della cella (part feeders, macchine di testing, attrezzi…) che richiedono non sono disgiunti, o se comunque esistono altri vincoli di layout. Tali operazioni si dicono allora *incompatibili*.

In questo capitolo, faremo sempre riferimento al caso in cui la relazione di precedenza tra le operazioni è di ordinamento totale: in altre parole, l'albero delle operazioni è una *catena*. Tale situazione è estremamente comune nelle lavorazioni meccaniche; peraltro, esso comprende anche quei processi di assiematura seriali per i

quali non si richiedono specifiche operazioni di predisposizione dei componenti elementari; quando invece esse possono incidere sul tempo di completamento della singola unità allora la topologia dell'albero di assiematura deve essere più generale (*pettine*, §3.2).

Dopo una breve revisione della (scarna) bibliografia esistente su problemi di gestione delle operazioni in celle robotizzate, nel §2.3 verranno richiamati alcuni concetti relativi ad un particolare problema di scheduling che fornisce il paradigma modellistico per i vari problemi di sincronizzazione affrontati in questo capitolo. Nel §2.4 vedremo il caso di una cella costituita da r robot e che è in grado di operare contemporaneamente su s unità. Gli utensili vengono allocati staticamente ai robot, ovvero non viene cambiata l'allocazione durante il funzionamento del sistema. La cella è progettata per produrre periodicamente un solo tipo di prodotto. Nel §2.5, invece, vedremo un'architettura in cui gli utensili possono essere spostati da un centro di lavorazione all'altro per mezzo di un sistema costituito da un robot e due navette. L'obiettivo, in ambedue gli scenari, è quello di massimizzare la produttività della cella. Si noti che sono possibili scenari intermedi (Han, Na e Hogg 1990), anche se non verranno analizzati in questo testo.

2.2 Letteratura sui problemi di gestione delle operazioni in FMC

La letteratura relativa ai problemi decisionali nelle celle flessibili non è molto ricca, e si possono per ora menzionare solo lavori che in realtà affrontano problemi decisionali alquanto diversi. Pressoché assenti sono comunque modelli per l'assegnamento delle operazioni del tipo di quelli analizzati nel seguito del §2; qualcosa in più esiste su problemi di sincronizzazione.

Blazewicz, Sethi e Sriskandarajah (1989) analizzano un problema di scheduling dei movimenti di un robot in una cella flessibile costuita da un solo robot e con diversi pezzi presenti nel sistema allo stesso tempo; il problema affrontato è, in pratica, di volta in volta quale dei pezzi nel sistema è più conveniente servire. Strutture diverse sono state analizzate da Wilhelm e Sarin (1985), che presentano una formulazione di programmazione matematica per lo stesso problema di sequenziamento dei movimenti del robot, mentre Kim e Langston (1987) hanno analizzato il problema di spostare un insieme di robot su un singolo binario da una configurazione iniziale a una finale nel tempo minimo.

Chen e Guerrero (1991) adottano un approccio basato su regole (tipico di molti problemi di intelligenza artificiale) per affrontare il problema di determinare i movimenti di un robot in una cella molto simile a quella analizzata da Blazewicz,

Sethi e Sriskandarajah. Come in quel caso, infatti, il robot si muove tra tre macchine e in pratica la sua unica funzione è di muovere i pezzi su cui avvengono le lavorazioni. Il loro approccio consiste nella costruzione di un grafo in cui i nodi corrispondono a tutti i possibili stati del sistema e gli archi alle possibili transizioni. Un ciclo su tale grafo corrisponde a una strategia di schedulazione per i robot, e un sistema esperto, sulla base del carico di lavoro richiesto dai pezzi che devono essere successivamente lavorati, sceglie di volta in volta la strategia più conveniente. Accanto alla maggiore flessibilità di un tale approccio rispetto a un modello combinatorio, lascia però perplessi la necessità di enumerare un numero di stati in generale molto elevato, anche per celle abbastanza semplici.

Reyman (1987) analizza il problema nell'ambito di un sistema costituito da molte celle, ognuna delle quali è in grado di operare indipendentemente dalle altre; il problema di sequenziare le operazioni in ogni cella è formulato in modi diversi, asseconda della struttura e del numero dei robot in ciascuna cella, sempre in termini di modelli di scheduling classici.

Han, Na e Hogg (1990) affrontano un problema di assegnamento di utensili che ha diversi punti in comune con quello presentato nel prossimo §2.5. Infatti, nel loro caso considerano un sistema costituito da vari centri di lavorazione, e ognuno di essi ha un magazzino utensili di limitata capacità a bordo macchina; quelli che non ha li può però ricevere, tramite un sistema di movimentazione utensili, da un magazzino centrale o in prestito da altri centri. Il problema consiste nel determinare quali utensili assegnare in modo statico a ciascuna macchina, con l'obiettivo di minimizzare l'entità complessiva del traffico nel sistema. Il problema è formulato come programmazione nonlineare a numeri interi e risolto in modo euristico.

2.3 Scheduling di due lavori con incompatibilità

In questo paragrafo richiamiamo alcuni concetti relativi ad un problema che nasce, come sottoproblema, nelle procedure risolutive dei problemi descritti nei §§2.4 e 2.5.

2.3.1 Descrizione del problema

Consideriamo una cella con due robot, i quali devono eseguire, rispettivamente, un job *A* e un job *B*. I due job consistono di una sequenza data di p e q operazioni rispettivamente. Sia A_i (B_j) la i–esima operazione di *A* (la j-esima di *B*). La durata dell'operazione A_i (B_j) sia $t(A_i)$ ($t(B_j)$). Inoltre, è data una lista di coppie di operazioni incompatibili. Uno *schedule* dei due job è un assegnamento di istanti d'inizio alle operazioni, ossia un vettore S con $p+q$ componenti $S=\{S(A_1), S(A_2),..., S(A_p), S(B_1),$

$S(B_2)$,..., $S(B_q)$} dove $S(A_i)$ ($S(B_j)$) denota l'istante d'inizio di A_i (B_j). Uno schedule tale che, ad ogni istante, le due operazioni attive sono compatibili, è detto *ammissibile*. Il tempo di completamento *T(A;B)* è dato dal *max* {$S(A_p) + t(A_p)$; $S(B_q) + t(B_q)$}.

Problema di Sincronizzazione di 2 job (2JSP)

Dati
due job *A* e *B*

trovare
uno schedule ammissibile

tale che
T(A;B) sia minimo

2JSP può risolversi per mezzo delle stesse tecniche usate per risolvere il problema di JOB SHOP con due lavori. Tale problema è stato per la prima volta studiato da Akers e Friedman (1955), il cui approccio non era però polinomiale. Succesivamente, Szwarc (1960) e Hardgrave e Nemhauser (1963) hanno proposto algoritmi polinomiali. L'algoritmo più efficiente che si conosca oggi è quello dovuto a Brucker (1988), che è stato confrontato anche con l'approccio da noi proposto.

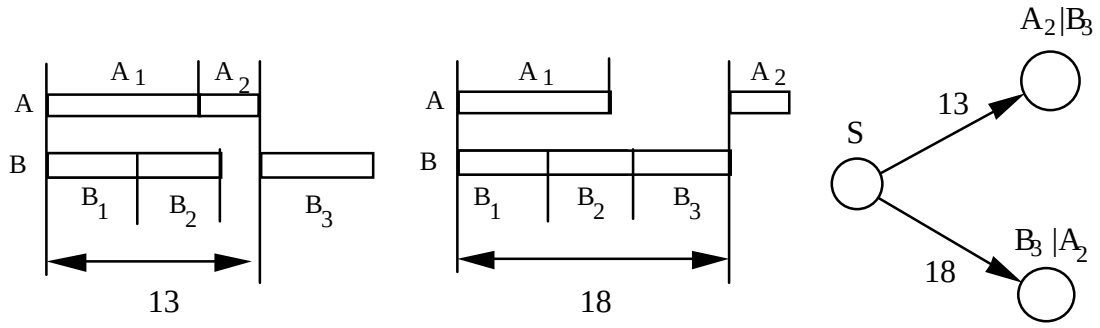


Figura 2.1. Conflitto tra le operazioni A_2 e B_3 e i due modi di risolverlo.

Nel seguito, sia $A^{(i)}$ ($B^{(j)}$) la sottosequenza consistente nelle sole operazioni A_i, A_{i+1},..., A_p (B_j, B_{j+1},..., B_q). Il problema *2JSP* può essere facilmente risolto come segue. All'inizio, i due job *A* e *B* sono eseguiti in parallelo finché si incontra la prima coppia di operazioni incompatibili, siano esse A_i e B_j. Dopodiché, o B_j deve attendere la fine di A_i o viceversa. Fatta una scelta, chiamiamo *milestone* l'istante di completamento della prima delle due operazioni in conflitto. Se A_i è eseguita prima di B_j, la distanza temporale dall'inizio fino al milestone è $t(A_1) + t(A_2) +...+ t(A_i)$, nell'altro caso è $t(B_1) + t(B_2) +...+ t(B_j)$. Rappresentiamo allora queste due alternative

per mezzo di due nodi, etichettati con $A_i|B_j$ e $B_j|A_i$ rispettivamente, e aggiungiamo un nodo S (che rappresenta l'inizio) connesso a ciascuno di questi due nodi. I due archi sono pesati con la corrispondente distanza temporale (Fig.2.1). Da ognuna delle due situazioni precedenti, possiamo poi proseguire, considerando il milestone relativo al nodo $A_i|B_j$ (al nodo $B_j|A_i$) come il punto d'inizio di un sottoproblema in cui i due job sono $A^{(i+1)}$ e $B^{(j)}$ ($A^{(i)}$ e $B^{(j+1)}$). Continuando così, si ottiene un grafo del tipo raffigurato in Fig.2.2, in cui gli archi sono pesati con la distanza tra due milestone, eccetto quelli entranti nel nodo P, che sono invece pesati con la distanza dal milestone alla fine dello schedule. Si noti che ogni nodo ha alpiù due archi uscenti. Il cammino minimo su questo grafo dà lo schedule ottimo.

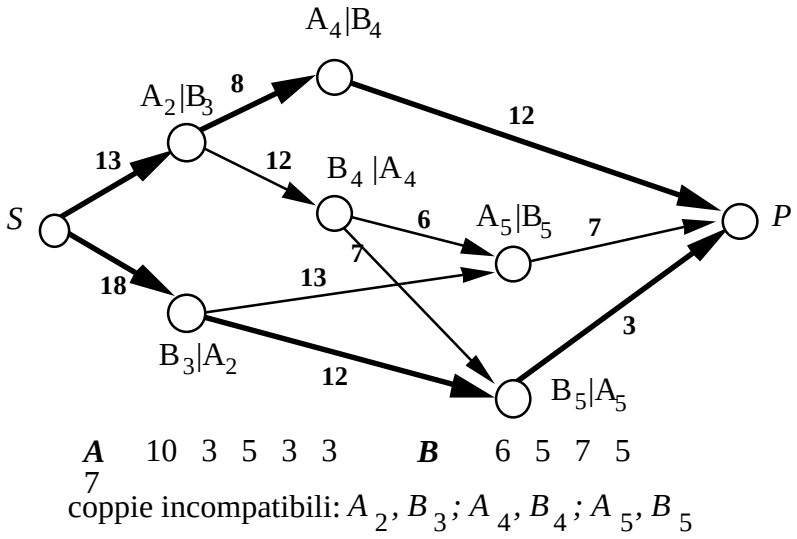


Figura 2.2. Il grafo Q : in grassetto due percorsi minimi.

	A		B	
1	70	μ	60	α
2	70	β	30	β
3	30	ε	70	μ
4	70	μ	70	ε
5	70	α	30	α
6	30	β	80	μ

Tabella 2.1. Esempio.

Secondo questa procedura, il grafo può essere generato in $O(n^3)$, dal momento che vi possono essere in generale $O(n^2)$ milestone (e quindi nodi) e il peso di ogni arco può essere calcolato in tempo lineare. Brucker (1988) propone una tecnica che porta alla generazione di un grafo con un maggior numero di nodi, in generale, ma in cui il peso di ogni arco può essere calcolato in tempo costante; il suo algoritmo richiede un tempo $O(n^2 \log n)$.

Per paragonare l'efficienza dei due approcci, sono state condotte delle prove numeriche nel caso medio. Chiamando ω la percentuale di coppie incompatibili sulle n^2 coppie, si è osservato che per valori realistici di ω (ossia, = 40%), il nostro approccio ha sempre dato luogo a tempi inferiori, nonostante la complessità asintotica sia peggiore (Agnetis e Oriolo 1993). Questo fatto sembra estremamente interessante, dal momento che la soluzione dei problemi che vedremo nei paragrafi successivi richiedono, in generale, la soluzione di un gran numero di istanze di *2JSP* (§2.4.3.4).

2.3.2 Rappresentazione grafica del problema

Nella letteratura riguardante il problema del JOB SHOP con due job, viene impiegata una rappresentazione grafica del problema che risulta di notevole utilità, anche per il problema che affronteremo nel §2.5.5. Consideriamo l'esempio in Tab.2.1: i due job *A* e *B* consistono di 6 operazioni ciascuno. Per ciascuna operazione è indicata la durata e, con una lettera greca, un utensile, a indicare che due operazioni che fanno uso dello stesso utensile sono incompatibili (ad es., A_2 e B_2).

Si consideri ora il piano *x-y*, ove gli assi *x* e *y* corrispondono ai job *A e B* rispettivamente (Fig.2.3). Le operazioni sono rappresentate da segmenti sui rispettivi assi, di lunghezza corrispondente alla loro durata, ordinati secondo le relazioni di precedenza tra esse. Sia Ψ la regione rettangolare che ha come vertici opposti $O(0,0)$ e $Q(\Sigma_i t(A_i), \Sigma_j t(B_j))$. Considerando le linee orizzontali e verticali che passano per gli estremi di ogni segmento, queste disegnano in Ψ una griglia di n^2 rettangoli, cioè uno per ogni coppia di operazioni (A_i, B_j). Se A_i e B_j sono incompatibili, la regione corrispondente è *proibita* (in Fig.2.3(*a*) le zone proibite sono ombreggiate). Un *cammino ammissibile* da *O* a *Q* è tale che: (*i*) consiste di una sequenza di segmenti orizzontali, verticali o diagonali a 45º, e (*ii*) non attraversa l'interno di nessuna regione proibita. Ad ogni cammino ammissibile si può far corrispondere una soluzione ammissibile al problema di sincronizzazione, come segue: i tratti diagonali corrispondono a intervalli di tempo in cui le lavorazioni su ambedue i job possono procedere, in quanto le operazioni attive sono compatibili; i tratti orizzontali (verticali) corrispondono invece a intervalli in cui solo il primo (il secondo) job è attivo. Definiamo allora come *lunghezza* di un cammino la lunghezza totale dei suoi tratti orizzontali più quelli verticali, che dunque rappresentano il tempo totale di attesa dei due centri, nell'ambito del periodo di lavorazione.

A questo punto, dal momento che minimizzare il tempo di completamento equivale a minimizzare le attese dei due centri, il problema di sincronizzazione è equivalente a trovare il *percorso ammissibile di lunghezza minima* da *O* a *Q*. In Fig.2.3 è indicato il percorso di lunghezza minima e la temporizzazione

corrispondente. Si noti che i segmenti orizzontali e verticali si hanno solo in corrispondenza di regioni proibite, che devono essere quindi aggirate: nella temporizzazione, questo corrisponde a un conflitto tra due operazioni incompatibili.

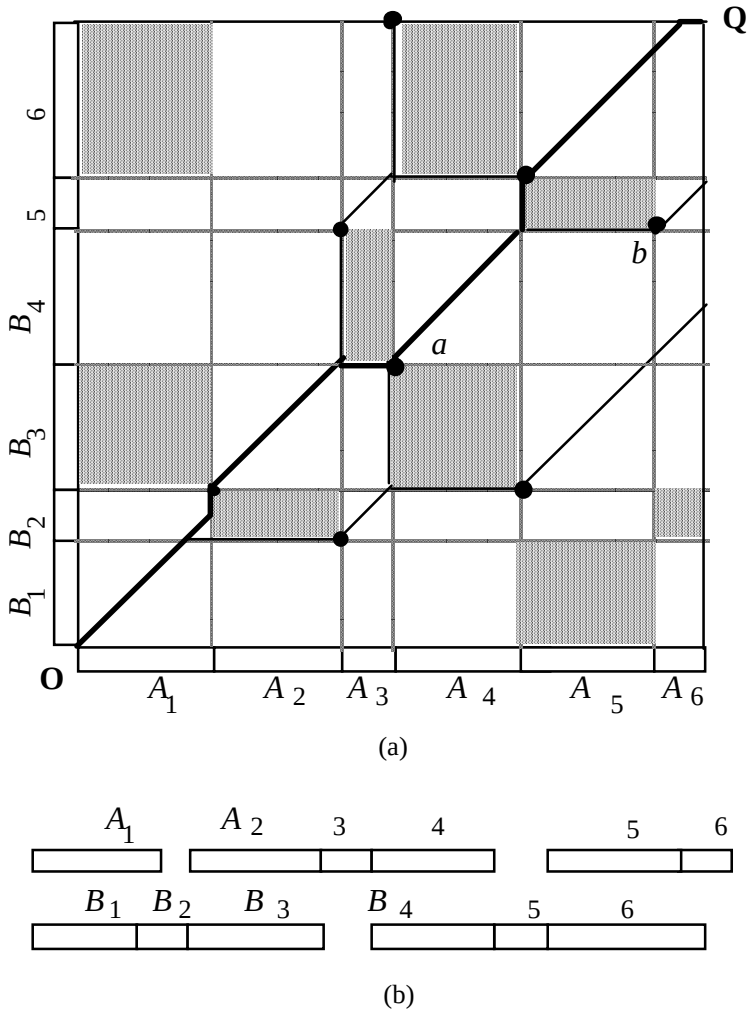


Figura 2.3. Rappresentazione grafica del problema di sincronizzazione in Tab.2.3
(a) La griglia Ψ (b) Temporizzazione corrispondente al percorso ottimo (in grassetto).

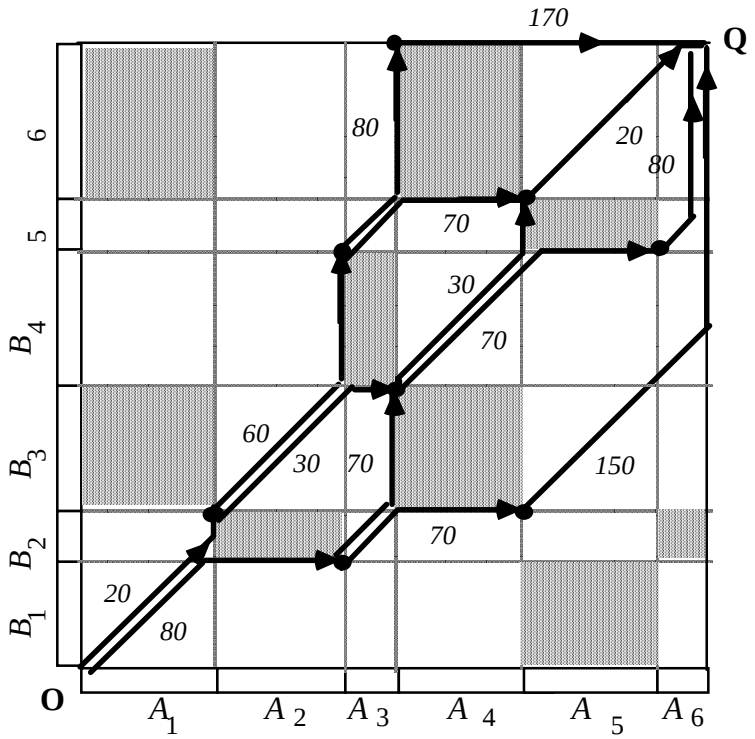
Benché tale problema sia stato definito con riferimento al piano *x-y*, il problema di percorso minimo può pensarsi come definito su un grafo generale, allorché si associno i nodi ai vertici di nord-ovest e di sud-est delle zone proibite, e gli archi a due vertici "visibili" reciprocamente (Fig.2.4). Il peso di ciascun arco sarà pari alla lunghezza del tratto di cammino orizzontale o verticale tra i due nodi.

2.4 Sincronizzazione di operazioni e assegnamento degli utensili in una cella per lavorazioni seriali

2.4.1 Introduzione

In questo capitolo, indicheremo le unità operatrici col termine *robot*, con riferimento

alle realizzazioni più comuni di celle flessibili. In linea di principio, una stessa operazione può essere eseguita da diversi robot, purché siano dotati di un set di risorse opportuno (utensili, fixtures, software…). Il costo di queste risorse può però essere alquanto elevato: si vuole allora assegnare ognuna di queste risorse ad un numero ristretto di robot. In una cella, tipicamente ciò che si fa è assegnare ognuna di queste risorse ad un *unico* robot, anche se ve ne possono essere diversi in grado di compierla. Tali risorse vengono dunque allocate *staticamente* ai robot, e nasce dunque il problema del modo più conveniente di effettuare tale allocazione. Per semplicità, chiameremo *utensili* questo tipo di risorse, anche se in realtà può trattarsi di oggetti di vario tipo: ad esempio, il programma di controllo del robot necessario a compiere un'operazione. Assegnando tale software in esclusiva a un robot, si evitano i costi legati alla sua duplicazione, e dunque allo sviluppo di programmi diversi.



(a)

Figura 2.4. Grafo associato a Ψ.

Se facciamo l'ipotesi che ogni operazione faccia uso di (almeno un) utensile, si noti che, dato un assegnamento di utensili a robot, viene indotto un unico routing nel sistema, che tutti i pezzi entranti seguiranno. Questa situazione è abbastanza simile alle linee di flusso flessibili o pipeline (§3.2), mentre è profondamente diversa da quella che si ha invece nei sistemi flessibili di produzione (§4), in cui invece la natura del sistema è tale da consentire una maggiore flessibilità di instradamento che può

essere sfruttata per bilanciare i carichi di lavoro tra le macchine. Infatti, in quel caso i problemi decisionali che nascono sono sostanzialmente diversi.

Concettualmente, l'assegnamento di utensili ai robot di un FMC ha un significato analogo a quello dell'attrezzaggio (*tooling*) in un FMS; tuttavia osserviamo che la differenza sta nel fatto che in un FMC l'assegnamento va fatto sulla base dell'interazione che avverrà tra i robot e le altre risorse del sistema, ed appare impossibile scindere tale problema da quello della sincronizzazione delle risorse stesse; nel caso degli FMS, viceversa, si fa uso di obiettivi più aggregati (§4.3), anche perché la complessità del sistema è tale che ben difficilmente si potrebbe tener conto dei profili temporali di dettaglio delle singole macchine.

Una prerogativa delle celle flessibili è quella di consentire di operare in parallelo su diverse unità (che supporremo sempre dello stesso tipo). Infatti, benché le operazioni relative ad ogni unità siano totalmente ordinate, per ogni macchina le operazioni relative a unità diverse devono essere opportunamente sequenziate e deve essere determinato l'istante d'inizio di ciascuna delle operazioni ad esse assegnate.

In questo capitolo ci occuperemo del problema *integrato* di (*i*) *assegnare* gli utensili (e di conseguenza le operazioni) alle macchine e di (*ii*) *sincronizzare* le operazioni. L'obiettivo è di massimizzare la produttività. Il problema nella sua generalità può essere molto complicato; è perciò utile sfruttare qualunque peculiarità che possa essere offerta da particolari classi di applicazioni. Una semplificazione notevole si ha se vale il cosiddetto vincolo di *esecuzione consecutiva*: ogni qual volta un robot esegue una sottosequenza di operazioni $O_i,\ldots, O_j$ (con O_{i-1} e O_{j+1} assegnate ad altri robot) su un'unità, è vincolato ad eseguirla *subito* anche sulle altre unità presenti nel sistema, prima di iniziare nuove sequenze di operazioni. Tale vincolo è legato alla capacità limitata dei magazzini utensili "on-board" (§2.4.2). Il problema risultante prenderà il nome di *PROC* (*Problema di Routing in una Cella flessibile*).

In quanto segue, r è il numero di robot, mentre s è il numero di unità che in ogni istante di tempo possono coesistere nel sistema (i.e., il numero di *stream*, §2.4.2). Il capitolo è organizzato così: nel §2.4.2 *PROC* è definito in modo più formale; nel §2.4.3 viene presentato un algoritmo polinomiale per *PROC* con s=2 e r=2 e si accennerà ad alcune possibili generalizzazioni che ne preservano la polinomialità; nel §2.4.5 verrà infine brevemente discussa la complessità di *PROC* in alcuni casi per cui r=3, rilassando (in vari modi) le condizioni che valevano nei precedenti paragrafi. I dettagli di alcune dimostrazioni possono trovarsi in (Agnetis, Lucertini e Nicolò 1993); un'applicazione modellata come un'istanza di *PROC* è presentata in (Agnetis, Lucertini e Nicolò 1991).

2.4.2 Definizioni e formulazione del problema

2.4.2.1 *Robot, utensili e stream*

Il processamento di ogni unità richiede l'esecuzione di un insieme di operazioni $O = \{O_1, O_2, \ldots, O_n\}$; tra le operazioni sussiste una relazione di ordinamento totale. La cella comprende *r robot* $\{R_1, R_2, \ldots, R_r\}$ e un insieme $U = \{u_1, u_2, \ldots, u_t\}$ di utensili che devono essere assegnati ai robot perché possano compiere le varie operazioni. Ad ogni operazione è associato un certo utensile; gli utensili sono disponibili in singola copia. Risulterà comodo talora riferirsi anziché all'insieme *O*, al *ciclo C* ottenuto dalla sequenza *O* chiusa su se stessa: $C = (O_i, O_{i+1}, \ldots, O_n, O_1, \ldots, O_{i-1})$. La durata di O_i è τ_i. Ogni robot, se possiede l'utensile opportuno, può eseguire qualsiasi operazione, con la stessa velocità (questa assunzione può in effetti essere facilmente rilassata, vedi (Agnetis, Lucertini e Nicolò 1993)). Le operazioni non possono essere interrotte e ogni robot può compiere alpiù un'operazione per volta; è dato l'insieme di coppie di operazioni incompatibili.

Alpiù *s* unità possono coesistere nella cella allo stesso tempo, principalmente a causa di limitazioni nel sistema di movimentazione. Chiaramente, in ogni istante robot diversi operano su unità diverse e le operazioni attive devono essere compatibili. Benché la sequenza di operazioni sia la stessa per tutte le unità che entrano nella cella, le *s* unità presenti a un generico istante seguiranno, in generale, *schedule* differenti, ossia, i tempi d'inizio di ciascuna operazione (rispetto all'istante in cui l'unità ha fatto il suo ingresso nella cella) possono essere diversi tra le varie unità (Fig.2.3). Dopo che un'unità esce dal sistema, una nuova unità entra, seguendo lo stesso schedule dell'unità che è appena uscita. Un insieme di unità che seguono tutte lo stesso schedule costituisce uno *stream*: unità degli *s* stream entrano nel sistema in ordine ciclico, e le *s* unità che coesistono nel sistema in un generico istante appartengono sempre a stream diversi. La lunghezza dell'intervallo di tempo che intercorre tra gli istanti d'inizio della stessa operazione su due unità consecutive *dello stesso stream* prende il nome di *periodo di produzione*, e verrà indicato con *T*. Supponiamo che *T* sia costante su ogni stream e sia uguale per gli *s* stream. La quantità *s/T* rappresenta il numero di unità prodotte nell'unità di tempo, ossia il *throughput* della cella. Dunque, per un fissato *s*, la massimizzazione del throughput (ossia della produttività) corrisponde alla minimizzazione del periodo di produzione.

Come conseguenza della discussione di cui sopra, l'intero processo produttivo è ciclico di periodo *T*, e possiamo limitarci ad analizzare lo schedule all'interno di una singola finestra temporale di lunghezza *T*. Senza perdita di generalità, possiamo prendere come istante 0 quello in cui inizia l'operazione O_1 sul primo stream. Si

consideri un'unità del k-simo stream, e sia Θ_{ki} il tempo d'inizio di O_i rispetto all'istante 0. Uno schedule è allora completamente specificato dal vettore $\{\Theta\}$.

Benché in molti casi $r=s$, ciò può non essere vero. Se $r>s$, vi sono sempre almeno $(r–s)$ robot in attesa; tuttavia, il throughput ottenibile in questa situazione è più alto che non se $r=s$. D'altro canto, se $s>r$, ci sono sempre almeno $(s–r)$ unità in attesa, ma l'utilizzazione dei robot è più elevata. Nel §2.4.4 analizzeremo la complessità di alcuni di questi casi "asimmetrici".

Ogni utensile deve essere assegnato esattamente ad un robot, per cui le operazioni che richiedono quell'utensile devono essere eseguite da uno *stesso* robot su tutti gli stream. Quest'esigenza, come si è già detto, si ha quando gli utensili sono costosi, e dunque una sola copia per utensile è disponibile. Dunque, una volta assegnati gli utensili, lo sono anche le operazioni.

2.4.2.2 *Il vincolo di esecuzione consecutiva*

Chiamiamo *sottocatena* una sequenza di operazioni consecutive nel ciclo C. Più specificamente, il simbolo (i,j) indica: la sottocatena $(O_i, O_{i+1},\ldots,O_j)$ se $i<j$, la sottocatena $(O_i, O_{i+1},\ldots, O_n,O_1,\ldots,O_j)$ se $j<i$. Il simbolo (i,i) o semplicemente (i) indica una sottocatena consistente della sola operazione O_i . Il numero totale di sottocatene è quindi n^2. Una sottocatena S_b è *consecutiva* a una sottocatena S_a se $S_a \leftrightarrow S_b = \emptyset$ e la prima operazione di S_b è adiacente all'ultima di S_a in C (così, se l'ultima operazione di S_a è O_n, la prima di S_b dev'essere O_1), ed è indicata con $S_a \varnothing S_b$. Si noti che se $S_a=(i,j)$ e $S_b=(j+1,i–1)$, vale sia $S_a \varnothing S_b$ che $S_b \varnothing S_a$. Dicendo che una sottocatena (p,q) è assegnata a un dato robot, intendiamo che il robot deve eseguire le operazioni da O_p a O_q, ma *non* $O_{p–1}$ e O_{q+1}.

L'assegnamento degli utensili ai robot deve tener conto della struttura dei dispositivi di manipolazione degli utensili: ogni robot ha un magazzino utensili a due livelli, diversi per la velocità di accesso e di cambio–utensile: abbiamo un magazzino ad *accesso rapido* (ad esempio, un mandrino multi–utensile), in cui sono tenuti gli utensili necessari per la prossima sequenza di operazioni, e un magazzino ad *accesso lento*. Quando un robot inizia l'esecuzione di una sottocatena S_i su un'unità, deve avere nel suo magazzino ad accesso rapido gli utensili richiesti dalle operazioni in S_i. Allorché deve essere eseguita una nuova sottocatena, è necessario un *set–up* per portare i nuovi utensili nel magazzino ad accesso rapido. Siccome in generale la capacità del magazzino ad accesso rapido è piccola, il robot non può eseguire una sequenza di operazioni arbitrariamente lunga senza set-up in mezzo. Allora, al fine di mantenere piccola la somma dei tempi di set–up (cosicché possano essere trascurati),

introduciamo un vincolo che implicitamente limita il numero complessivo di set–up. La sottocatena S_i si dice *ammissibile* se le operazioni in S_i possono essere eseguite senza set-up intermedi (nel seguito, per brevità, il termine *ammissibile* spesso verà omesso). Siccome ogni robot esegue le sue operazioni su tutte le unità, in ogni periodo di produzione può dover caricare *s* volte gli utensili richiesti dallo stesso set di operazioni. Quindi, quando un robot inizia l'esecuzione delle operazioni di una sottocatena S_i, appare ragionevole far sì che il robot le esegua *consecutivamente su tutte le unità*, eliminando così (*s*–1) set-up per ogni sottocatena. Così si realizza un considerevole risparmio di tempo e appare giustificato trascurare i tempi di set–up. Più precisamente, il vincolo di *esecuzione consecutiva* può definirsi come segue:

– *Sia S_i=(p,q) una sottocatena ammissibile assegnata al robot R_j. Dopo l'esecuzione di O_p su una unità, R_j deve completare tutte le operazioni di S_i su tutte le altre unità presenti nel sistema, prima di iniziare qualunque altra operazione*.

oper	utens.	τ_i	incomp	**oper**	utens.	τ_i	incomp
$\boldsymbol{O_1}$	u_1	6	O_4	$\boldsymbol{O_4}$	u_4	4	O_1
$\boldsymbol{O_2}$	u_2	3	—	$\boldsymbol{O_5}$	u_1	5	—
$\boldsymbol{O_3}$	u_3	1	—	$\boldsymbol{O_6}$	u_3	6	—

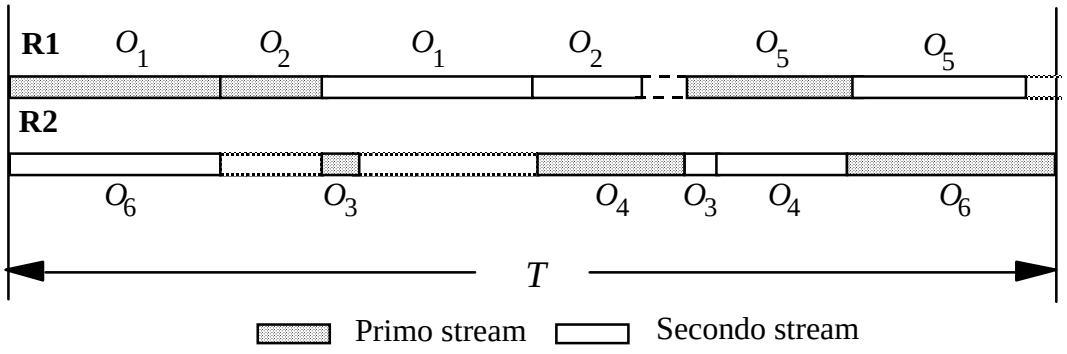


Figura 2.5. Esempio.

2.4.2.3 *Formulazione del problema*

In definitiva, il problema *PROC* può formularsi così:

***Problema di Routing seriale in Celle flessibili (PRoC)*:**

Dati

il numero *s* di stream e il numero *r* di robot; l'insieme *O* di operazioni con le loro durate $\{\tau_i\}$ (i=1,2,…,n), e l'insieme $U=\{u_1, u_2,\ldots, u_t\}$ di utensili; la lista di coppie di operazioni incompatibili;

---

trovare
l'assegnamento di utensili a robot e lo schedule $\{\Theta_{iq}\}$ (i=1,2,…,n; q=1,2,…,s)
tali che
il throughput sia massimizzato, ovvero che il periodo di produzione T sia minimizzato e
il vincolo di esecuzione consecutiva sia soddisfatto

Nel seguito, T*indicherà il valore ottimo di T.

In molte applicazioni, la struttura del problema verifica un'ulteriore condizione, che, come vedremo nei §2.4.3 e §2.4.4, influisce fortemente sulla complessità del problema:

***Condizione δ**. Tutte le operazioni richiedono utensili diversi.*

Se tale condizione è soddisfatta, si può chiaramente parlare indifferentemente di utensili e operazioni.

La Fig.2.5 illustra un esempio ed una soluzione ammissibile di *PROC*. In questo esempio, r=2, s=2, n=6. Le due barre orizzontali corrispondono ai due robot, mentre i due stream sono indicati da due ombreggiature differenti. La durata delle operazioni, gli utensili richiesti da ciascuna operazione e la lista delle incompatibilità sono riportate nella tavola. Si noti che in questo esempio la condizione δ non è verificata. Nella soluzione proposta, gli utensili u_1 e u_2 (e quindi le operazioni O_1, O_2 and O_5) sono assegnati a R_1, gli altri a R_2. La durata del peridoo di produzione è T =30. Si noti che lo schedule rispetto all'istante iniziale di O_1 è diverso per i due stream: ad esempio, le unità del primo stream attendono un certo tempo tra O_3 e O_4, mentre per le unità del secondo stream O_4 inizia subito dopo la fine di O_3.

Per brevità, nei paragrafi a venire useremo la notazione *PROC(s,r)* per indicare *PROC* con s stream e r robot. Vedremo dapprima un algoritmo polinomiale per *PROC(2,2)* con la condizione δ ; successivamente vedremo la complessità di *PROC* in situazioni diverse: s=2, r=3; s=3, r=3 con condizione δ.

2.4.3 Soluzione di PROC(2,2) con condizione δ

In questo paragrafo, viene descritta una procedura polinomiale di soluzione per *PROC(2,2)*. In questo paragrafo supponiamo che la condizione δ (§2.4.2) sia soddisfatta. Ciò ci consentirà dunque di identificare operazioni e utensili. Nel seguito, se l'indice y di un'operazione è più grande di n, indica l'operazione O_{y-n}.

Dapprima verrà analizzata la struttura delle soluzioni ottime di *PROC(2,2)*, quindi vedremo un teorema che consente il calcolo di una soluzione ottima.

2.4.3.1 *Struttura delle soluzioni ottime di PROC(2,2)*

Sia Π una partizione delle operazioni del ciclo C in $2k$ sottocatene ammissibili $\{S_1, S_2,\ldots, S_{2k}\}$. Supponiamo che le operazioni appartenenti alle sottocatene dispari $S_1, S_3,\ldots,S_{2k-1}$ siano assegnate a R_1, e quelle appartenenti alle sottocatene pari $S_2, S_4,\ldots, S_{2k}$ a R_2. Chiaramente, Π definisce un assegnamento di utensili a robot. Ricordando la formulazione di *PROC*, cerchiamo una partizione Π^* e uno schedule $\{\Theta\}$ tale da minimizzare il periodo di produzione T.

Il problema ha una struttura "nidificata", che consente di trattare separatamente con i due aspetti del problema, vale a dire quello relativo all'assegnamento delle operazioni da un lato e quello relativo alla sincronizzazione delle operazioni, dall'altro. Il seguente semplice lemma (per la cui dimostrazione si veda (Agnetis, Lucertini e Nicolò 1993)), esprime una proprietà delle soluzioni ottime di *PROC(2,2)*. Per esercizio, si mostri la differenza tra la definizione di vincolo di esecuzione consecutiva, e quanto asserito dal lemma.

Lemma 1. *Sia S_i una sottocatena assignata a R_j nell'assegnamento ottimo (j=1,2). Esiste uno schedule ottimo Θ^* in cui R_j esegue prima l'intera sottocatena S_i senza altre operazioni in mezzo su un'unità e poi ripete la stessa sottocatena sull'altra unità.*

Come conseguenza di questo lemma, la struttura di una soluzione ottima di *PROC(2,2)* è quella raffigurata in Fig.2.6, in cui riconosciamo $2k$=4 *sottoschedule*. In un sottoschedule, le operazioni della sottocatena S_i sono eseguite in parallelo a quelle di S_{i-1} o S_{i+1}. La decomposizione del periodo di produzione in un numero pari di sottoschedules è una proprietà generale delle soluzioni ottime di *PROC(2,2)* (perché?).

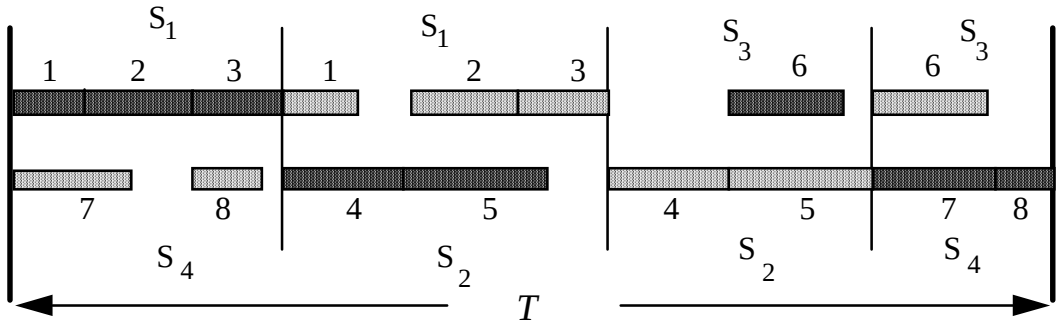


Figura 2.6. Una solutione ammissibile per PROC(2,2).

Si noti che il modo in cui le operazioni sono sincronizzate in un sottoschedule *è del tutto indipendente* da quello relativo ad altri sottoschedule. Siccome il nostro obiettivo è minimizzare il periodo di produzione, in ogni sottoschedule le operazioni saranno sincronizzate in modo tale da minimizzare la lunghezza di quel sottoschedule. Quindi se, in un sottoschedule, il robot R_1 è incaricato dell'esecuzione di una sottocatena S_i, e R_2 di S_{i+1}, il sottoschedule ottimo resulta dalla soluzione di un'istanza di *2JSP* (§2.3), in cui il job *A* corrisponde alla sottocatena S_i e il job *B* alla sottocatena S_{i+1}. Ovviamente, le durate delle operazioni sono quelle date. In conclusione, se l'assegnamento ottimo Π di operazioni a robot fosse noto, il problema si ridurrebbe semplicemente a risolvere 2*k* istanze di *2JSP*. Nei paragrafi che seguono, analizzeremo perciò il problema di trovare l'assegnamento ottimo Π^*.

2.4.3.2 *Un algoritmo polinomiale per PROC(2,2)*

Vogliamo di seguito mostrare che *PROC(2,2)* può formularsi come problema di determinare un opportuno ciclo su un grafo orientato.

Sia *G=(N,A,W)* un grafo orientato pesato, in cui i nodi corrispondono a sottoschedule e ogni arco unisce due sottoschedules che possono essere consecutivi in una soluzione ammissibile del problema. Precisamente, definiamo l'insieme di archi e di nodi come segue:

– Per ogni coppia di sottocatene consecutive $S_a \varnothing S_b$ vi sono due nodi, indicati come $S_a||S_b$ e $S_b||S_a$ (la notazione $S_a||S_b$ indica che S_a è eseguito da R_1 e S_b in parallelo da R_2);

– Date tre sottocatene consecutive $S_a \varnothing S_b \varnothing S_c$, c'è un arco dal nodo $S_a||S_b$ al nodo $S_c||S_b$ e uno dal nodo $S_b||S_a$ a $S_b||S_c$.

Se $S_a+(i,j)$ e $S_b+(j+1,i-1)$, è sia $S_a \varnothing S_b$ che $S_b \varnothing S_a$. Quindi, date tre sottocatene consecutive $S_a \varnothing S_b \varnothing S_c$, può essere S_a+S_c: in tal caso vi sono *due nodi* $S_a||S_b$, uniti da due archi (a formare un ciclo) e disconnessi da tutto il resto del grafo, e, allo stesso modo, *due nodi* $S_b||S_a$.

I *nodi* sono pesati nel seguente modo: data una coppia di sottocatene consecutive, siano esse *(i,j)* e *(j+1,h)*, consideriamo l'istanza di *2JSP* ottenuta prendendo le due sottocatene come job. Il valore del tempo di completamento ottimo di questa istanza dà il peso dei nodi *(i,j)||(j+1,h)* e *(j+1,h)||(i,j)*. In altre parole, ogni nodo corrisponde a un sottoschedule ed è pesato con la durata del sottoschedule. In quanto segue, il peso del nodo $S_i||S_j$ sarà indicato con $T(S_i||S_j)$.

Il grafo G consiste di O(n^3) nodi e O(n^4) archi. Se definiamo due insiemi di nodi N_1 e N_2 come

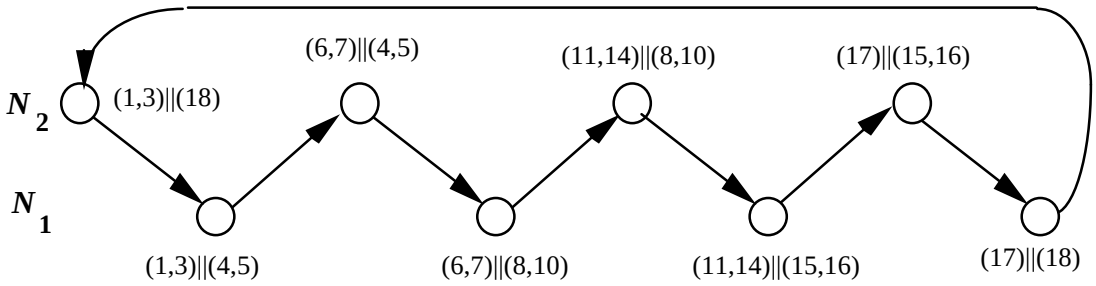
$$N_1 = \{ \text{ nodi } S_a \| S_b \text{ per cui } S_a \varnothing S_b \}$$

$$N_2 = \{ \text{ nodi } S_a \| S_b \text{ per cui } S_b \varnothing S_a \}$$

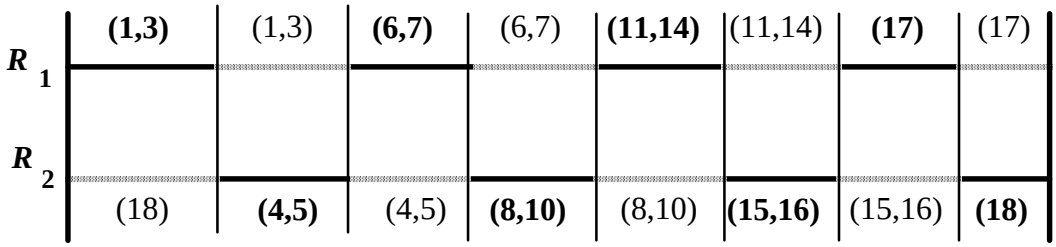
osserviamo che *G è bipartito*, dal momento che tutti gli archi vanno da N_1 a N_2 o viceversa: quindi, tutti i cicli hanno un numero pari di archi (Fig.2.7(a)).

Data una soluzione di *PROC(2,2)*, è possibile associarvi un ciclo sul grafo *G*, identificando il campo di sinistra con una sottocatena assegnata a R_1, e quello di destra con una sottocatena assegnata a R_2 (vedi Fig.2.7). Non tutti i cicli su *G* corrispondono però a soluzioni ammisssibili di *PROC(2,2)*. Non lo è, ad esempio, in un'istanza in cui n = 6, un ciclo come (1)||(2,4), (5,6)||(2,4), (5,6)||(1), (2,4)||(1), (2,4)||(5,6), (1)||(5,6). Si noti che in questo caso, però, viene violato il vincolo per cui ogni operazione deve essere assegnata esclusivamente a un robot. Diciamo allora che un ciclo su *G* è *semplice* se ogni operazione compare esattamente in due nodi (consecutivi). Esiste una corrispondenza uno-a-uno tra cicli semplici sul grafo *G* e le soluzioni del problema *PROC(2,2)*, ossia le partizioni Π del ciclo *C*. In Fig.2.7(a), i nodi in basso (in alto) appartengono a N_1 (N_2). Il valore del periodo di produzione è pari alla somma dei pesi dei nodi. Dunque, rimane dimostrato il seguente teorema:

Teorema 2.1 – *Una soluzione ottima di PROC(2,2) è data dallo schedule corrispondente al ciclo semplice di lunghezza minima in G.*



(a)



(b)

Figura 2.7. Corrispondenza tra (a) cicli semplici di *G* e (b) soluzioni di *PROC*(2,2).
Lo scheduling di dettaglio non è indicato. *C* si compone di 18 operazioni.

2.4.3.3 Costruzione del grafo G: complessità

Analizziamo la complessità dei calcoli richiesti per la costruzione di *G*, i.e., dal calcolo del peso di tutti i nodi. *G* consiste di O(n^3) nodi, e per ogni nodo occorre risolvere un'istanza di *2JSP*. D'altro canto, consideriamo il seguente gruppo di O(n) nodi:

$$(i,j)||(j+1);\ (i,j)||(j+1,j+2);\ (i,j)||(j+1,j+3)\ ;\ \ldots\ ;\ (i,j)||(j+1,j+q) \qquad (1)$$

Questi nodi corrispondono a tutti i sottoschedule ammissibili in cui R_1 esegue la sottocatena *(i,j)* su uno stream, mentre R_2 esegue la successiva sottocatena sull'altro stream. La sottocatena *(j+1,j+q)* è la più lunga sottocatena ammissibile che inizia con O_{j+1} che un robot può eseguire (eventualmente, gli indici $j+q$ e $i-1$ possono coincidere).

E' facile verificare che tutti i sottoproblemi corrispondenti a questi nodi possono essere risolti tramite un'unica istanza di *2JSP*. Per convincersi di questo, ricordiamo l'approccio diretto presentato nel §2.3. Si consideri un'istanza di *2JSP* in cui associamo la sottocatena *(i,j)* al job *A* e la sottocatena *(j+1,j+q)* al job *B*. E' possibile costruire il grafo *Q* in modo da ottenere anche i pesi degli altri nodi in (1), senza ulteriore appesantimento computazionale. Dobbiamo solo inserire q nodi-"target" P_1, P_2,…, P_q con grado di uscita zero, uno per ogni sottoschedule *(i,j)||(j+1,j+k)*, con

1=k=q. Ogni nodo target P_k corrisponde al nodo finale di un'istanza di *2JSP* in cui il job B è troncato alla (j+k)-esima operazione. Si vede facilmente che la complessità del calcolo dei pesi di tutti questi nuovi archi (che uniscono i nodi di Q ai nodi target) non supera O(n^3), e quindi tale rimane la complessità del calcolo di *tutti* i percorsi minimi su Q. siccome ci sono $O(n^2)$ gruppi di nodi come (1) in G, la complessità globale della costruzione di G è $O(n^5)$, seguendo il metodo delineato nel §2.3. Si potrebbe mostrare che usando l'approccio di Brucker questo limite può in effetti essere abbassato a $O(n^4 \log n)$. In pratica tuttavia la convenienza relativa dei due metodi non è così ovvia (fine del §2.3).

2.4.3.4 *Calcolo del ciclo semplice di peso minimo: algoritmo e complessità*

Per il calcolo del ciclo semplice di lunghezza minima, in quanto segue, tutte le disuguaglianze vanno lette in senso ciclico. Per il Lemma 1, dato l'assegnamento ottimo Π^*, ogni sottocatena S_i Π^* è prima eseguita interamente su uno stream, e poi sull'altro. Perciò, indichiamo gli stream come *primo* e *secondo*, a seconda dell'ordine in cui subiscono le operazioni di ogni sottocatena.

Supponiamo che una *data* sottocatena (u,v) sia stata assegnata a R_1, e sia z(u,v) il valore di una soluzione ottima di *PROC(2,2)* con questo vincolo ulteriore. Mostreremo ora un modo efficiente di calcolare z(u,v); dopodiché, considerando tutte le possibili sottocatene (u,v), il miglior valore di z(u,v) sarà scelto come z*. Le variabili che introduciamo nel seguito sono tutte relative al problema con (u,v) assegnato a R_1; per semplicità notazionale, tuttavia, omettiamo u e v dai nomi delle variabili. Non andremo a ragionare esplicitamente sul grafo G, anche se tutto ciò che diremo è del tutto equivalente. In particolare, poiché supporremo inizialmente che una *data* sottocatena (u,v) sia stata assegnata a R_1, ciò che vedremo equivale a rendere aciclico il grafo G eliminando tutti i nodi in cui una delle due sottocatene abbia intersezione non nulla con (u,v), e a inserire un nodo-sorgente e un nodo-pozzo. Il nodo-sorgente è connesso a tutti i nodi del tipo (u,v)||(v+1); (u,v)||(v+1,v+2); (u,v)||(v+1,v+3);… , mentre il nodo-pozzo ha come predecessori tutti i nodi del tipo (v–1)||(u,v); (v–2,v–1)||(u,v); (v–3,v–1)||(u,v);… Si noti che un *cammino* dal nodo sorgente al nodo pozzo su questo nuovo grafo corrisponde esattamente a un *ciclo semplice* su G (una volta assegnata la sottocatena (u,v) a R_1).

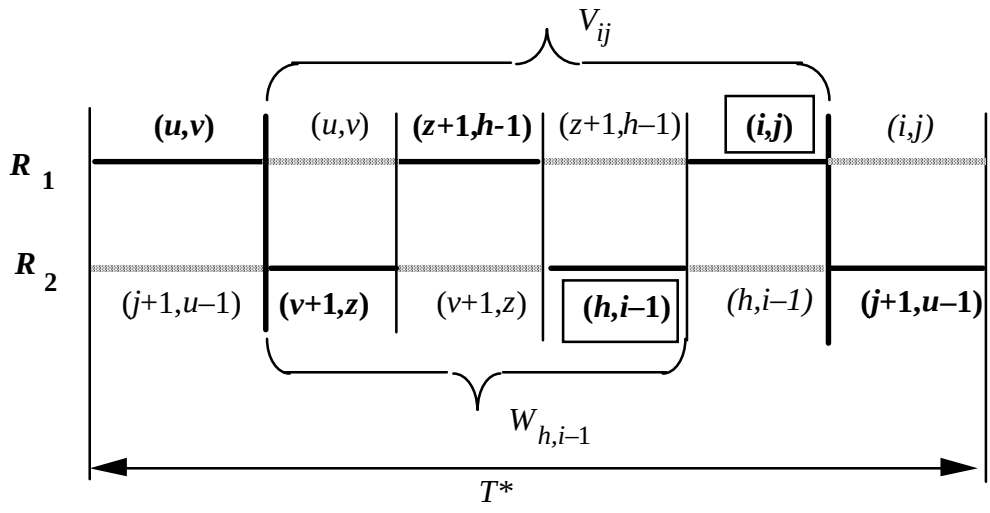


Figura 2.8. Definizione di V_{ij} e $W_{h,i-1}$. I due stream sono indicati con diverse ombreggiature.

Consideriamo una partizione Π che includa, tra le sue sottocatene, la sottocatena (u,v), e che perciò ha una struttura del tipo:

$$\{(u,v), (v+1,z), \ldots, (i,j), (j+1, u-1)\}$$

come visto, il valore di T associato a questa partizione è ottenuto sommando le durate dei singoli sottoschedule. Consideriamo una coppia di operazioni O_i, O_j tale che $v<i<j<u$, e sia V_{ij} la minima lunghezza di una *sequenza* di un numero *pari* di sottoschedule, tale che nel primo sottoschedule la sottocatena (u,v) è eseguita sul secondo stream e nell'ultimo sottoschedule la sottocatena (i,j) è eseguita sul primo stream. Sia poi W_{ij} definito allo stesso modo, ma con un numero *dispari* di sottoschedule (Fig.2.8). E' sottinteso che i valori V_{ij} e W_{ij} sono definiti solo per sottocatene (i,j) ammissibili; negli altri casi li consideriamo pari a $+\infty$. Si noti che i valori V_{ij} e W_{ij} esprimono solo una *porzione* della lunghezza dell'intero periodo di produzione T, che si compone anche del tempo relativo ai due sottoschedule tramite i quali si richiude il ciclo sul grafo G.

Per calcolare i V_{ij}, osserviamo che

$$V_{v+1,w} = +\infty \qquad \text{per } w = v+1,\ldots \tag{2}$$

$$W_{v+1,w} = T(u,v\|v+1,w) \qquad \text{per } w = v+1,\ldots \tag{3}$$

infatti, se la sottocatena (u,v) è assegnata a R_1, l'operazione O_{v+1} è assegnata a R_2 (così, $(v+1,w)$ non può essere assegnata a R_1). In generale, si ha

$$V_{ij} = \min_{v+1 = k = i-1} [\ W_{k,i-1} + T(i,j\|k,i-1)\ ] \tag{4}$$

$$W_{ij} = \min_{v+1 = k = i-1} [\ V_{k,i-1} + T(k,i-1||i,j)\ ] \qquad (5)$$

e dunque, dapprima si calcolano tutti i valori del tipo $V_{v+2,\supseteq}$, quindi tutti i $W_{v+2,\supseteq}$; seguono poi i $V_{v+3,\supseteq}$; i $W_{v+3,\supseteq}$ e così via. Una volta calcolati tutti i V_{ij} e i W_{ij} , si può calcolare $z(u,v)$ che è dato da

$$z(u,v) = \min_{i,j} [\ V_{ij} + T(i,j||j+1,u-1) + T(u,v||j+1,u-1)\ ] \qquad (6)$$

e quindi, in conclusione:

$$T^* = \min_{u,v} \{\ z(u,v)\ \} \qquad (7)$$

da cui la partizione Π^* può essere facilmente ricostruita.

Per quanto concerne la complessità della procedura sopra descritta, osserviamo che il calcolo di ogni V_{ij} e ogni W_{ij} richiede un tempo O(n), e ci sono $O(n^2)$ valori V_{ij} e W_{ij}. Quindi, ogni $z(u,v)$ può calcolarsi in tempo $O(n^3)$. Siccome vi sono evidentemente $O(n^2)$ modi di scegliere (u,v), il calcolo del ciclo semplice di lunghezza minima può essere effettuato in tempo O(n^5). Ricordando che la fase di costruzione del grafo G (ossia, del calcolo di tutte le quantità $T(i,j||j+1,u-1)$) poteva essere svolta anch'essa in o(n^5), risulta dimostrato il seguente teorema:

Teorema 2.2 – *PROC(2,2) con la condizione δ può essere risolto in O(n^5).*

In conclusione, l'algoritmo risolutivo per *PROC(2,2)* può scriversi come segue:

Algorithm *SHORTEST_CYCLE* (***input***: ciclo *C*; ***output***: soluzione ottima di *PROC(2,2)*);

Begin
genera tutte le sottocatene ammissibili*(u,v)* ;
for ogni coppia S_a ,S_b di sottocatene ammissibili consecutive **do**
2-JOBS_SCHEDULING (S_a;S_b); $T(S_a||S_b) = T(S_b||S_a)$ *:=T*(S_a;S_b);*
for tutte le sottocatene *(u,v)* **do**
begin
for i=v+1,…, u–1 **do**
for j= i,…, u–1 **do**
calcola V_{ij} e W_{ij} con le (2)—(5) con (u,v) assegnato a R_1;
calcola $z(u,v)$ con la (6);
end;
T^* := $\min_{u,v} \{\ z(u,v)\ \}$

End *SHORTEST_CYCLE*.

2.4.4 Complessità di PROC

Nel §2.4.3 abbiamo visto che *PROC(2,2)* è polinomiale se vale la condizione δ. In questo paragrafo analizziamo la complessità in due casi diversi: prima supporremo s=2 e r=3 (§2.4.4.1), e quindi per s=3, r=3 (§2.4.4.2). In pratica, vedremo che il solo problema di assegnamento degli utensili è NP–completo, anche nei casi in cui la sincronizzazione è banale mentre, per contro, in generale *PROC* è NP-completo anche se l'assegnamento di utensili ai robot è banale.

2.4.4.1 *Complessità di PROC(2,r), r=3*

Analizziamo la complessità di *PROC(2,r)*. La condizione δ gioca un ruolo cruciale nella determinazione della complessità del problema: infatti, se essa è verificata, il problema è polinomiale, altrimenti è NP–completo.

Soffermiamoci su *PROC(2,3)*, e consideriamo il caso in cui δ è verificata. Allora, il problema può risolversi esattamente con lo stesso approcio presentato nel §2.4.3: siccome s=2 e vale il vincolo di esecuzione consecutiva, il Lemma 1 è ancora valido e le soluzioni ottime di *PROC(2,3)* sono ancora decomponibili in sottoschedule (Fig.2.9(a)). Siccome s=2, in ogni istante c'è sempre almeno un robot in attesa. Così, possiamo costruire un grafo orientato (indicato con H nel seguito) in cui nodi e archi hanno lo stesso significato che nel grafo G di §2.4.3: ogni nodo corrisponde a un sottoschedule e ogni arco unisce due sottoschedule potenzialmente consecutivi. Stavolta si noti che un sottoschedule è rappresentato da un nodo che ha *tre* campi; due di essi corrispondono a robot attivi, l'altro al robot in attesa. Per esempio, un sottoschedule in cui R_1 deve eseguire la sottocatena (i,j), R_2 la sottocatena $(j+1,k)$, ed R_3 è in attesa sarà indicato con $(i,j)||(j+1,k)||$xxx. In conclusione, i nodi e gli archi di H sono definiti come segue (con ovvio significato dei simboli):

– Per ogni coppia di sottocatene consecutive $S_a \varnothing S_b$ + $(i,j)\varnothing(j+1,k)$ sono definiti *sei* nodi, indicati con $(i,j)||(j+1,k)||$xxx, $(j+1,k)||(i,j)||$xxx, $(i,j)||$xxx$||(j+1,k)$, $(j+1,k)||$xxx$||(i,j)$, xxx$||(i,j)||(j+1,k)$ e xxx$||(j+1,k)||(i,j)$, tutti pesati con il tempo di completamento ottimo di un'istanza di *2JSP* in cui i due job sono le due sottocatene $(j+1,k)$ e (i,j).

– Date tre sottocatene consecutive $S_a \varnothing S_b \varnothing S_c$, c'è un arco da ogni nodo corrispondente a un sottoschedule in cui S_a e S_b sono eseguite in parallelo, a ogni nodo corrispondente a un sottoschedule in cui S_c e S_b sono eseguite in parallelo, e tale che S_b è eseguito dallo stesso robot in entrambi i sottoschedule.

La Figura 2.9 illustra la corrispondenza tra soluzioni ammissibili di *PROC* e cicli semplici sul grafo *H*. *H* consiste di O(n^3) nodi e O(n^4) archi; quindi il ciclo minimo su *H* può ancora trovarsi in tempo polinomiale. Si noti che, a differenza del caso in cui *s*=2 e *r*=2, una soluzione ottima può consistere di un numero dispari di sottoschedule: in altre parole, *H* non è bipartito (Fig.2.9(b)).

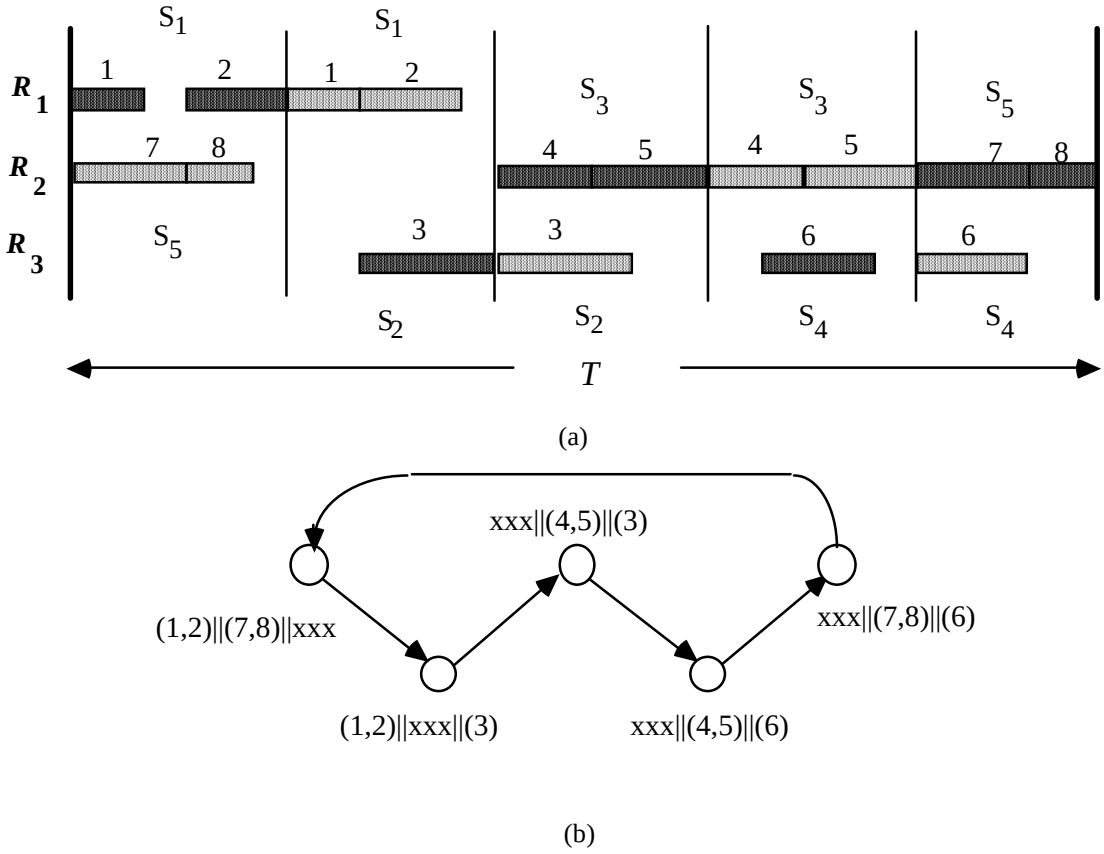


Figura 2.9. (a) Una soluzione ammissibile di *PROC(2,3)*. (b) Il corrispondente ciclo su *H*.

A questo punto dovrebbe essere chiaro che lo stesso approccio può essere usato con *r* robot e due stream: in questo caso generale possiamo associare, ad ogni coppia di sottocatene consecutive, *r* (*r*–1)/2 nodi, uno per ogni coppia di robot attivi. Può facilmente vedersi che *H* ha O(n^3r^2) nodi e O(n^4r^3) archi: infatti, ogni nodo è connesso con O(*nr*) altri nodi (quelli che hanno S_b nello stesso campo e tutti i possibili S_c consecutivi in uno degli altri *r*–1 campi).

Se ora rilassiamo la condizione *δ*, *PROC(2,3)* diventa NP-completo. La dimostrazione avviene per riduzione da GRAPH 3–COLORABILITY:

"*Dato un grafo G(V,E), è 3-colorabile?* ".

La dimostrazione è un po' laboriosa dal punto di vista tecnico, ma l'idea è abbastanza semplice: a partire da un'istanza di GRAPH 3–COLORABILITY, viene costruita

un'istanza di *PROC* in cui esiste un modo di assegnare gli utensili ai tre robot in modo che il periodo di produzione sia pari al minimo teorico (ossia, non vi siano attese da parte dei pezzi), se e solo se il grafo originario ammetteva una 3–colorazione. Si dimostra cioè che (Agnetis, Lucertini e Nicolò 1993):

Teorema 2.3 – *PROC(2,3) è NP-completo, anche se* $\tau_i=1$ *per ogni* $i=1,2,\ldots,n$.

2.4.4.2 *Complessità di PROC(s,r) con r=3 e s=3*

Infine, se sia *r* che *s* sono maggiori di 2, *PROC* diviene NP-completo, anche se vale la condizione δ. Il risultato non dovrebbe sorprendere più di tanto, dal momento che è stato recentemente dimostrato che JOB SHOP con 3 job è NP-completo (Sotskov 1992). In pratica, basta associare ai tre job tre stream, e definire opportunamente le operazioni della catena di operazioni di *PROC*. La dimostrazione è anch'essa riportata in (Agnetis, Lucertini e Nicolò 1993).

2.5 Gestione dinamica degli utensili in una cella costituita da due centri di lavorazione

2.5.1 Introduzione

In questo capitolo analizzeremo un problema di gestione delle operazioni che si presenta in una cella flessibile costituita da due centri di lavorazione. Rispetto a quella discussa nel precedente capitolo, l'architettura della cella differisce per i seguenti aspetti:

— le unità operatrici (nel seguito indicati come *centri di lavorazione*) non sono mobili: ciascun pezzo viene montato su un centro all'inizio del proprio ciclo di lavorazione e non ne viene più tolto fino a ultimazione delle operazioni;

— gli utensili non sono assegnati in modo statico ai centri di lavorazione, bensì vengono trasportati da un punto all'altro del sistema da un sistema di movimentazione utensili, costituito da un *robot* e due *navette*.

Il problema decisionale che analizziamo in questo capitolo è quello di sincronizzare i movimenti di robot e navette in modo da massimizzare l'utilizzazione dei centri di lavorazione. In generale, la situazione che si presenta in una cella di lavorazione è quella di un insieme di pezzi che devono essere prodotti in certe quantità, o periodicamente in dati rapporti relativi (del tipo di quelli che vengono calcolati, per un sistema di tipo diverso, nel §3.4.3). In ogni caso, un problema a monte della sincronizzazione riguarda l'allocazione dei pezzi ai centri (dal momento che una volta

montati su un centro, esso esegue tutte le operazioni) e il loro sequenziamento. Problemi di questo tipo appaiono estremamente complessi (è quasi immediato ridurre problemi NP–completi ad essi), e l'unico metodo praticamente utilizzabile è fare uso di algoritmi euristici, tipicamente di ricerca locale. Proprio allo scopo di valutare la bontà di una soluzione ammissibile, nell'ambito di un algoritmo di ricerca locale, nasce allora il problema di sincronizzare le operazioni in modo ottimo. Come già nel §2.4, infatti, anche qui due operazioni possono essere incompatibili; precisamente, ciò accade se richiedono lo stesso utensile, ed esso è presente in copia singola nel sistema. Vedremo nel §2.5.4 che il problema di determinare la migliore sincronizzazione degli utensili può risolversi per mezzo di un approccio che generalizza quello, visto nel §2.3, per il *2JSP*. Per semplicità nel seguito ci riferiremo al caso in cui a ogni centro di lavorazione sono assegnati un numero finito di pezzi, in sequenza data: l'input del nostro problema è allora pressoché identico a quello di un'istanza di *2JSP*, in quanto possiamo associare ai due centri di lavorazione due job, ciascuno costituito dalla sequenza di tutti i pezzi assegnati a quel centro. La differenza rispetto al problema *2JSP* sta nel fatto che, a causa della struttura della cella e dei tempi di spostamento di robot e navette, possono sorgere diversi tipi di conflitto tra diverse operazioni (§2.5.4).

La terminologia usata in questo capitolo è mutuata da un sistema reale, per cui si osservi che il termine *robot* ha qui un significato sostanzialmente diverso da quello che aveva nel §2.4. Nel §2.5.2 viene meglio illustrata la struttura del sistema in esame, nel §2.5.3 viene illustrato l'algoritmo risolutivo.

2.5.2 La cella

La cella oggetto della presente analisi è schematizzata in Fig.2.10. Nelle lavorazioni viene utilizzato un insieme U di utensili. Tali utensili possono essere in numero alquanto elevato (anche parecchie centinaia), e sono utilizzati dai centri di lavorazione per operare sui pezzi. Per ogni pezzo da produrre, è definito un insieme di operazioni, per le quali valgono le ipotesi:

— l'ordine in cui i pezzi devono essere processati da ogni macchina è prefissato, e siccome per ogni pezzo l'insieme delle operazioni è totalmente ordinato, ogni centro di lavorazione deve eseguire una prefissata sequenza di operazioni;

— ogni operazione O_i ha una certa *durata* e richiede l'impiego di un *utensile*;

— due operazioni sono *incompatibili* se e solo se richiedono lo stesso utensile, e questo è presente in copia singola nel sistema.

Si osservi che quest'ultima ipotesi è lievemente più restrittiva rispetto a quanto visto nel modello del §2.4, ove l'incompatibilità tra due operazioni poteva essere causata da qualunque motivo. Tuttavia essa corrisponde a quello che accade in pratica in celle flessibili del tipo in Fig.2.10, in cui gli spazi di lavoro delle unità operatrici non si intersecano, ma la loro interazione avviene attraverso lo scambio di utensili. Nel seguito siamo interessati a considerare solo quegli utensili presenti in copia singola, in quanto sono gli unici che possono generare incompatibilità.

In molti casi applicativi, tra cui le lavorazioni meccaniche, la catena di lavorazione relativa a un pezzo tipicamente richiede un numero di utensili compreso tra 20 e 80, con valor medio 40 e varianza molto alta; è possibile che lo stesso utensile venga utilizzato più volte nell'ambito dello stesso pezzo, per operazioni distinte. Gli utensili complessivamente presenti nel sistema possono arrivare anche a 600. La durata delle singole operazioni di lavorazione può variare tra 1 minuto e 15 minuti, con distribuzione abbastanza uniforme in questo intervallo.

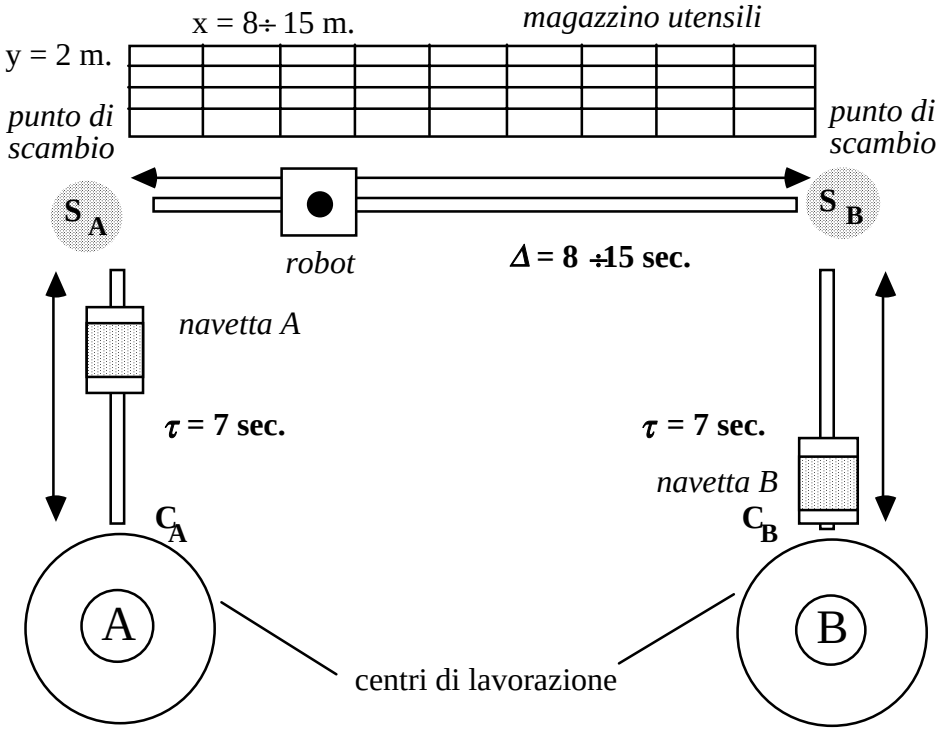


Figura 2.10. Rappresentazione schematica della cella in esame.

Dunque, i centri di lavorazione, per eseguire ciascuna operazione, devono usare un certo utensile. Siccome supponiamo che i centri di lavorazione non possiedano strutture per l'immagazzinamento di utensili, essi, quando non sono usati da qualche centro, sono allocati in un *magazzino centrale*, da cui vengono estratti e fatti pervenire ai centri a cura di un sistema di movimentazione, sincronizzandone

opportunamente gli invii.

Il *magazzino centrale* è concepito come uno scaffale (o rastrelliera) al quale ha accesso un sistema di "storage and retrieval" (nel seguito per brevità *robot*). In breve, il robot è in grado di spostarsi parallelamente all'armadio, e di estrarre o riporre un certo utensile da/in una precisa locazione. Il parametro di interesse per quel che concerne l'insieme *magazzino utensili + robot* è il tempo di spostamento del robot da un capo all'altro del magazzino, nel seguito indicato con Δ Il robot è in grado di trasportare un utensile alla volta.

Ogni utensile viene trasportato al centro di lavorazione per mezzo di una *navetta*, la quale anche è in grado di trasportare un utensile alla volta. Alla fine di una operazione sul centro di lavorazione, la navetta ha il compito di effettuare uno scambio utensili con il centro, prelevando l'utensile usato e caricando quello nuovo. L'utensile usato viene trasportato al robot, dal quale invece riceverà l'utensile necessario per la successiva lavorazione. Un parametro di interesse è il tempo che la navetta impiega a percorrere il tratto dal centro di lavorazione al punto in cui si interfaccia con il robot; tale tempo verrà indicato con τ.

2.5.3 La produzione e gli obiettivi

Massimizzare la produttività del sistema a fronte delle risorse disponibili corrisponde a minimizzare la durata del periodo di tempo entro il quale tutte le lavorazioni in esame devono essere completate. Il problema è allora quello relativo alla *sincronizzazione* delle lavorazioni, ossia si tratta di determinare, nell'ambito di un periodo di produzione, l'istante iniziale di ognuna delle operazioni di ogni pezzo, allo scopo di minimizzare la lunghezza del periodo di produzione. L'interesse dell'analisi è proprio legato al fatto che gli utensili sono costosi, e dunque si ha interesse a limitarne il numero: il fatto che certi utensili siano presenti in copia singola nel sistema, fa sì che certe operazioni non possono essere eseguite in parallelo: se i due centri richiedono uno stesso utensile contemporaneamente, occorrerà assegnarlo ad uno dei due: l'altro, perciò, dovrà attendere che il primo centro lo abbia rilasciato, e che robot e navette glielo facciano pervenire.

Come per il problema *2JSP*, indicheremo con A_k (B_h) la *k*–esima (*h*–esima) operazione che deve essere eseguita dal centro *A* (*B*). Introduciamo allora il seguente problema (*P*):

Sincronizzazione di due job in una cella flessibile (P)

Dati
Due sequenze di operazioni $A=\{A_1, A_2,\ldots, A_n\}$ e $B=\{B_1, B_2,\ldots, B_m\}$ che devono essere eseguite dai due centri di lavorazione;
Per ogni operazione A_i (B_j), la durata $t(A_i)$ ($t(B_j)$) e l'utensile richiesto $u(A_i)$ ($u(B_j)$);
I tempi τ e Δ di spostamento delle navette e del robot rispettivamente;

Determinare
Un assegnamento di istanti d'inizio a ogni operazione su ogni centro;
Uno schedule di navette e robot compatibile con tale assegnamento;

tale che
il tempo di completamento T_c delle operazioni delle due sequenze sia minimo.

Non consideriamo esplicitamente il tempo necessario a scambiare gli utensili tra navetta e centri di lavorazione: tale tempo può essere a tutti gli effetti inglobato in quello dell'operazione relativa. Nel prossimo §2.5.4 esso verrà risolto, in modo esatto, per mezzo di un efficiente modello di programmazione dinamica.

Osserviamo infine che il sistema preso in considerazione consiste di due soli centri di lavorazione. Da tre in su, gli stessi problemi di sincronizzazione discussi per due centri sono estremamente complessi: infatti, il problema di sincronizzazione diviene una generalizzazione del JOB SHOP con 3 job, che come già detto nel §2.4.4, è NP-completo.

2.5.4 L'algoritmo risolutivo per P

Il problema *P* risulta essere una generalizzazione del *2JSP*, che si ottiene per $\tau = \Delta = 0$. Se invece τ e Δ sono positivi, occorre analizzare cosa può accadere in presenza di un conflitto. Come si vedrà nel seguito, se due operazioni sono incompatibili, dopo che una delle due termina, l'altra, per iniziare, deve ancora attendere un po' di tempo (necessario a fargli pervenire l'utensile).

Nel seguito supporremo valida un'assunzione, che è di norma verificata in tutti i casi pratici: quando un centro di lavorazione richiede un utensile attualmente ubicato nel magazzino centrale, l'operazione corrispondente può iniziare senza ritardi, ossia è sempre possibile far pervenire al centro l'utensile prima dell'inizio dell'operazione stessa. Il modello presentato nel seguito vale dunque per valori generici dei parametri τ e Δ, ma purché questa condizione sia verificata.

Nel seguito, parleremo di *conflitto* allorché due operazioni (incompatibili) richiedono lo stesso utensile. L'algoritmo che verrà ora illustrato per l'assegnamento di istanti iniziali alle operazioni delle due sequenze, procede "da sinistra verso destra", ovvero allocando temporalmente le operazioni in ordine crescente di indice. Indichiamo con $S(A_i)$ $(S(B_j))$ l'istante d'inizio assegnato all'operazione A_i (B_j), e con $F(A_i)$ $(F(B_j))$ gli istanti finali, ossia $F(A_i) = S(A_i) + t(A_i)$ $(F(B_j) = S(B_j) + t(B_j))$.

2.5.4.1 *Conflitti semplici.*

Nel seguito, facciamo riferimento all'esempio in Fig.2.11. Come avveniva per *2JSP*, finché non si verificano conflitti, ogni operazione delle due sequenze può iniziare immediatamente dopo la fine della precedente.

Le due sequenze consistono di 7 operazioni. La durata di ciascuna operazione è indicata accanto all'utensile necessario per compierla (gli utensili sono identificati da lettere greche). Supponiamo che tutti gli utensili siano presenti in copia singola nel magazzino centrale. E' possibile eseguire A e B in parallelo fino alle operazioni A_1 e B_2. Dopodiché, l'esecuzione di A_2 costringerebbe a ritardare l'inizio dell'operazione B_3 di un tempo sufficiente a trasferire l'utensile γ da A a B. Ci troviamo allora davanti ad una situazione di conflitto tra le operazioni A_2 e B_3. A seconda che venga data la precedenza ad A_2 oppure a B_3, si ottengono le due sincronizzazioni in Fig.2.12: la lunghezza T_t dell'intervallo di tempo impiegato dall'utensile per "migrare" da un centro all'altro, chiaramente, è pari a $(2\tau+\Delta)$.

	A		B	
1	100	μ	90	λ
2	60	γ	80	δ
3	120	ε	60	γ
4	90	α	115	β
5	100	β	70	α
6	200	ψ	60	ψ
7	130	ω	80	ρ

τ = 7.5 sec.

Δ = 15 sec.

Figura 2.11. Esempio di problema *P*.

Osserviamo meglio la Fig.2.12(a): dopo che A ha rilasciato l'utensile γ, può chiaramente iniziare l'esecuzione dell'operazione A_3. Tale operazione si sovrappone evidentemente con B_2, ma questo fatto non pone problemi in quanto queste due operazioni sono compatibili (si noti che questo problema non si pone nell'altro caso, in quanto A è in attesa durante l'esecuzione di B_3). In altri termini, nell'"intorno" delle operazioni che danno luogo al conflitto, non ci sono altre operazioni incompatibili. In questo caso si parla allora di *conflitto semplice*.

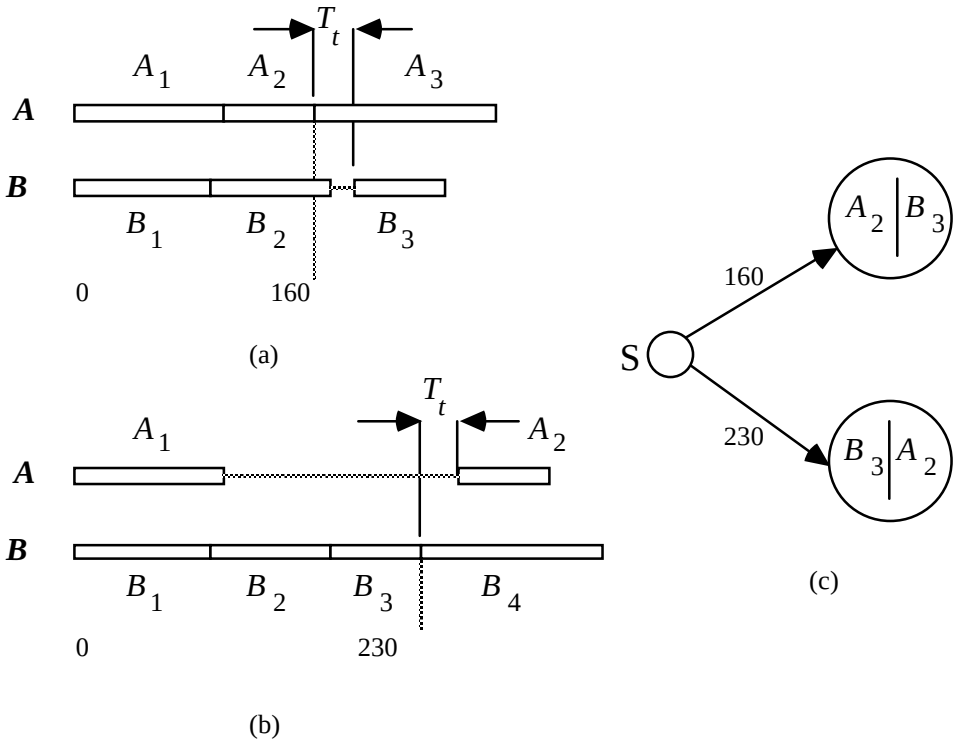


Figura 2.12. I due modi di risolvere il conflitto semplice tra A_2 e B_3.

Nel caso di conflitto semplice, analogamente a quanto avveniva in *2JSP*, ci troviamo evidentemente di fronte a due possibilità, indicate in Fig.2.12(a)-(b). Come per *2JSP*, vogliamo definire un punto nel tempo che abbia il significato di disaccoppiare l'evoluzione successiva dello schedule da quella precedente: indichiamo allora con *milestone* l'ascissa temporale alla quale finisce il primo dei due lavori in conflitto (si poteva anche optare per la convenzione di chiamare milestone l'istante a cui inizia il secondo dei due lavori). La distanza che intercorre dall'istante *t=0* al milestone dipende da quale delle due scelte è stata operata per risolvere il conflitto: abbiamo perciò, nell'esempio, 160 nel caso che γ sia assegnato prima ad *A* e 230 nell'altro caso.

Una volta operata una di queste due scelte, la sincronizzazione può essere proseguita normalmente, finché non si presentano nuovi conflitti. Al solito, per tenere traccia di tutte le possibili alternative che si presentano nella risoluzione dei conflitti, introduciamo un *grafo G*, in cui i nodi corrispondono ai modi di risolvere i conflitti – e quindi ai punti di disaccoppiamento – e gli archi alle distanze tra due di essi.

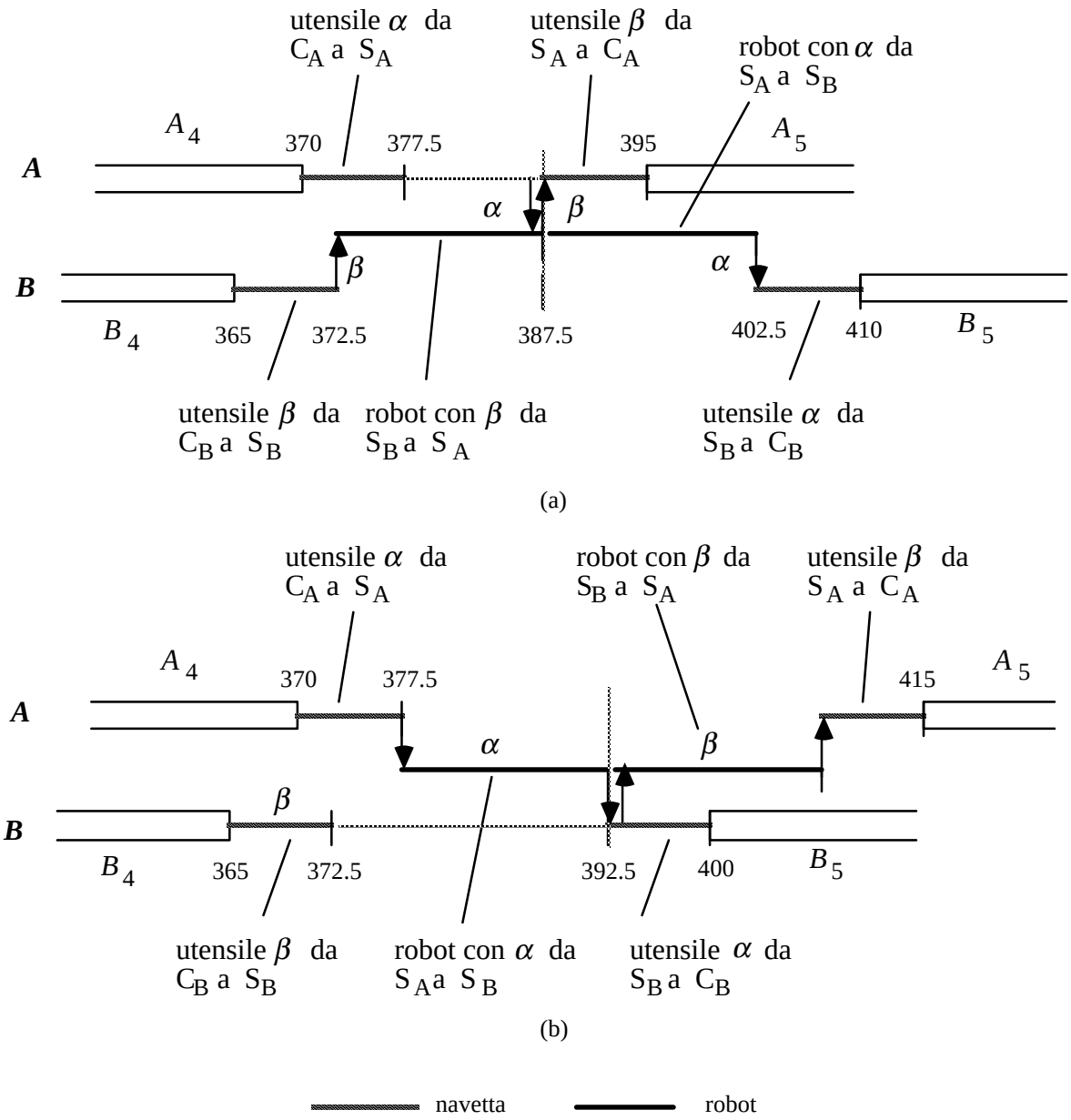


Figura 2.13. Modi di risolvere il conflitto doppio tra le coppie A_4,B_5 e A_5,B_4.
(a) Il robot raccoglie prima l'utensile β e poi α. (b) Il robot raccoglie prima l'utensile α e poi β.

Continuando l'esempio in Fig.2.11, nel caso in cui abbiamo dato la precedenza a B_3 rispetto ad A_2, è facile verificare che il successivo conflitto (semplice) avverrà tra le operazioni A_4 e B_5. Se invece viene data la precedenza ad A_2, si arriva ad una situazione in cui le operazioni B_4 e A_5 entrano in conflitto, in quanto richiedono ambedue l'utensile β. A differenza di quanto accadeva prima, però, si ha che se viene data la precedenza a B_4, l'operazione B_5 *non* può iniziare dopo un tempo $(2\tau+\Delta)$ dalla fine di B_4, in quanto l'utensile α è ancora utilizzato da A. In altri termini, il conflitto non è un conflitto semplice, e i modi in cui può essere risolto sono più di due.

2.5.4.2 *Conflitti doppi.*

Un conflitto semplice si verifica allorché un utensile deve essere trasferito da un centro all'altro, e questo provoca una attesa sul centro che viene servito per secondo. Nel caso che stiamo esaminando, invece, la situazione è più complicata, in quanto sono *due* gli utensili che devono essere trasferiti, precisamente α da A a B e β da B ad A. Parliamo allora di conflitto *doppio*. Vediamo quali alternative si prospettano, e come vanno definiti ora i milestone.

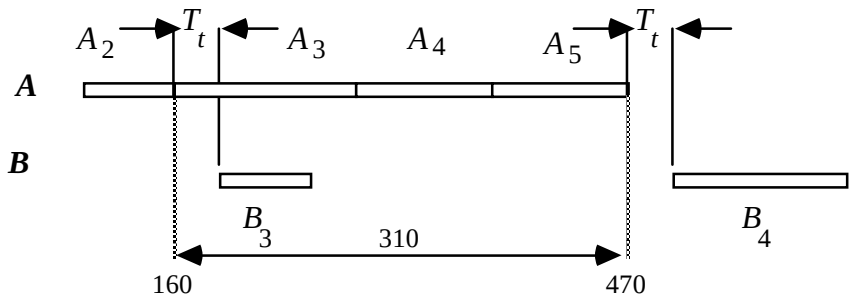
Un primo modo di risolvere il conflitto doppio è quello raffigurato in Fig.2.13(a). I due utensili sono stati qui assegnati alle due operazioni che ne fanno richiesta per prime: dunque, le operazioni A_4 e B_4 ricevono gli utensili α e β, e possono essere effettuate in parallelo. In questo caso, B_4 termina prima di A_4. Al termine di B_4, il robot si fa trovare in S_B, ove prende l'utensile β dalla navetta B, e inizia il suo tragitto verso S_A, dove arriva dopo un tempo Δ. Nel frattempo, la navetta A si è pure portata in S_A, con l'utensile α. Una volta in S_A, avviene lo scambio di utensili tra navetta A e robot; dopodiché il robot torna verso S_B ove ad attenderlo trova la navetta B, pronta a ricevere l'utensile α. Si noti che la risorsa in un certo senso "critica", in questo conflitto, è proprio il robot: infatti, le due navette — e di conseguenza i due centri — devono attendere i suoi spostamenti per scambiarsi gli utensili.

In Fig.2.13(a), il robot va a prendere l'utensile che si rende per primo disponibile, ossia β. Ancorché ragionevole, nulla ci garantisce che questa scelta sia quella ottima: infatti, osserviamo che se il robot si fa trovare in S_A dalla navetta A, l'istante di inizio di A_5 verrà certamente ritardato, ma quello di B_5 sarà invece anticipato. Di conseguenza, un altro modo di risolvere il conflitto che va considerato è quello raffigurato in Fig.2.13(b).

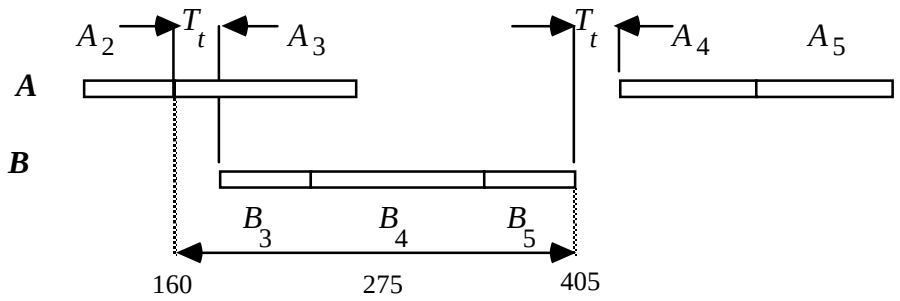
In ambedue i casi considerati, le operazioni facenti uso degli utensili α e β sono comunque svolte in parallelo, come pure (parzialmente) in parallelo avviene il trasferimento degli utensili. Abbiamo tuttavia altre due possibilità di risolvere il conflitto doppio: una scelta poteva essere quella di assegnare *entrambi gli utensili* α *e* β prima al centro A e poi al centro B. Siccome B ha bisogno di β per l'esecuzione di B_4, questa scelta implica che B è in uno stato di attesa finché A non rilascia β. Un discorso simmetrico vale, ovviamente, se si sceglieva di assegnare ambedue gli utensili prima a B. Le due situazioni sono raffigurate in Fig.2.14. Si noti che in questi ultimi due casi ci si è sostanzialmente ricondotti a considerare un conflitto semplice.

Qual è, nelle quattro situazioni considerate, il milestone? Dunque, occorre individuare un punto dopo il quale le successive scelte risultano indipendenti da quelle fatte in precedenza. Nei casi di Fig.2.14, analogamente a quanto visto per il conflitto semplice, tale punto può essere preso alla fine dell'operazione che rilascia

l'utensile conteso. In Fig.2.14 sono indicati tali istanti, come pure la loro distanza temporale dal precedente punto di disaccoppiamento. Nei due casi in Fig.2.13, l'evento disaccoppiante, per così dire, è rappresentato dallo scambio degli utensili che avviene, rispettivamente, in S_A e in S_B. Infatti, dopo tale istante abbiamo:



(a)



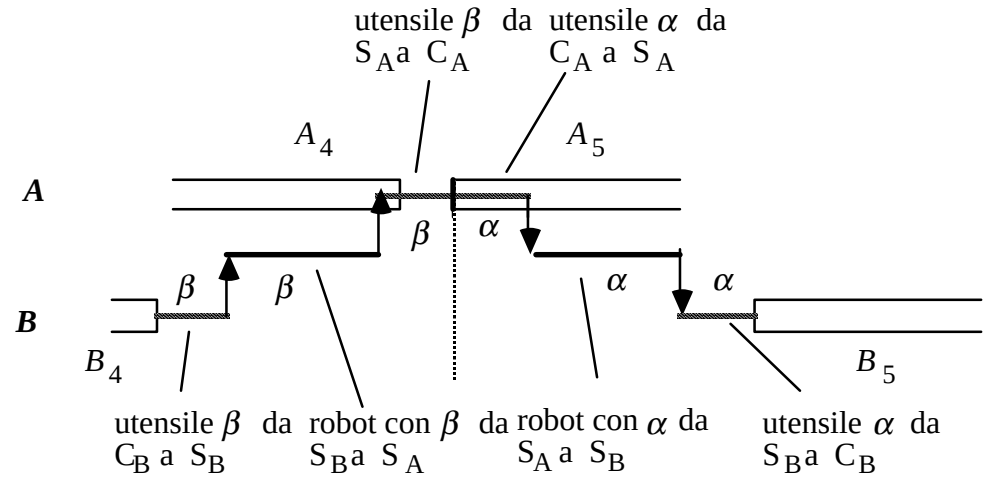
(b)

Figura 2.14. Gli altri due modi di risolvere il conflitto doppio. α e β sono assegnati (a) ambedue prima ad *A* (b) ambedue prima a *B*.

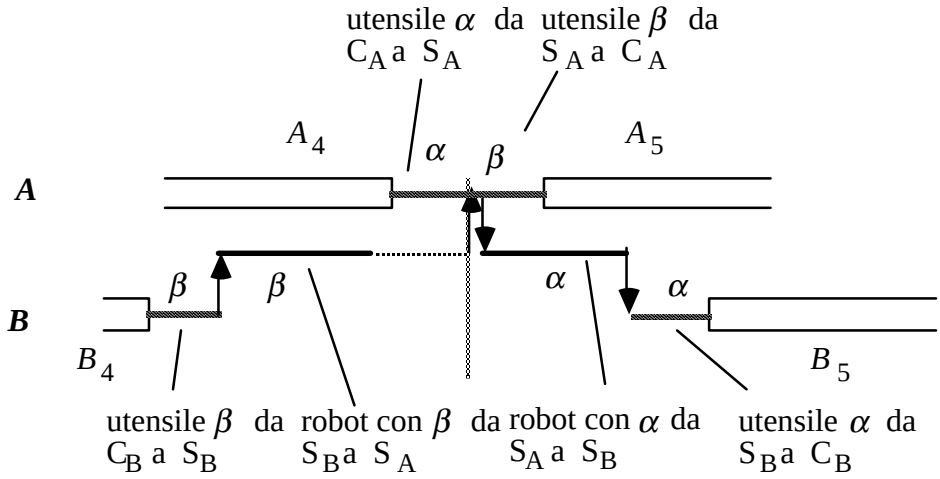
– nel caso di Fig.2.13(a), per il centro *A* un'attesa di τ, dopodiché può partire A_5, e per il centro *B* un'attesa di $\tau + \Delta$, dopo la quale può partire B_5;

– nel caso di Fig.2.13(b), per il centro *A* un'attesa di $\tau + \Delta$, dopodiché può partire A_5, e per il centro *B* un'attesa di τ, dopo la quale può partire B_5.

I milestone sono dunque, nei due casi in Fig.2.13, t=387.5 e t=392.5, e di conseguenza possono calcolarsi le corrispondenti distanze dal precedente punto di disaccoppiamento (227.5 e 232.5).



(a)

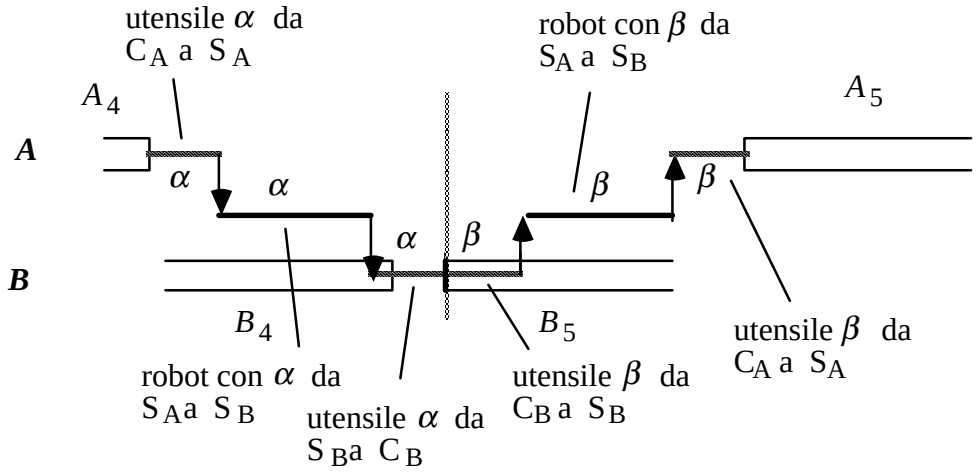


(b)

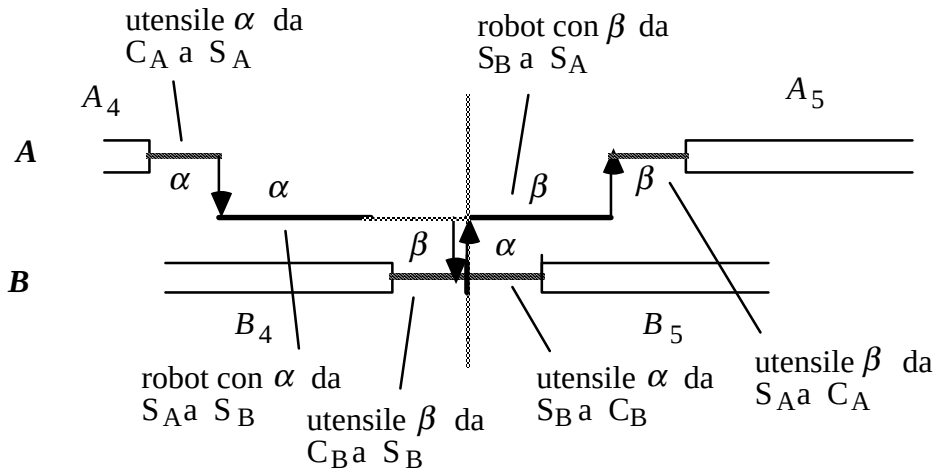
Figura 2.15. Un'altra possibile situazione di conflitto doppio.

Nell'esempio che stiamo sviluppando, giunti in presenza del conflitto doppio, abbiamo esaminato quattro modi di risolverlo. Essi sono effettivamente tutti e soli i modi "convenienti" (senza cioè introdurre *inutili* attese su ambo i centri) che andavano presi in considerazione. Si noti che nei casi illustrati in Fig.2.13 viene lasciato, "a valle" del milestone, un intervallo di attesa lungo τ su una macchina ed uno lungo $\tau + \Delta$ sull'altra. Ciò è conseguenza del fatto che, come già detto, la risorsa critica (ossia che determina tali attese) è il robot. Tuttavia, poteva anche presentarsi una situazione diversa, con uno sfasamento maggiore tra gli istanti finali di A_4 e B_4 (vedi Fig.2.15(a)): qui, mentre l'utensile β viene trasportato da B verso S_A, il centro A deve ancora terminare la sua operazione. Può allora essere conveniente far sì che, quando arriva il robot, la navetta A si trovi già in S_A, pronta a ricevere β e portarlo verso A. In questo caso, dopo il punto di disaccoppiamento, A_5 può immediatamente

iniziare, mentre B_5 può iniziare solo dopo un tempo $2\tau+\Delta$. In questo caso, la risorsa "critica" è la navetta *A*: l'utensile α non può partire immediatamente alla volta dell'altro centro perché la navetta *A* è impegnata a portare il nuovo utensile — si noti che se gli istanti finali di A_4 e B_4 distassero di più di $2\tau + \Delta$, il conflitto sarebbe semplice.



(a)



(b)

Figura 2.16. Due ulteriori possibilità per un conflitto doppio.

Tuttavia, analogamente agli altri casi, non è detto che convenga effettivamente dare la precedenza all'utensile β, nell'uso della navetta *A*: se si fosse optato per l'altro caso, avremmo ottenuto la temporizzazione in Fig.2.15(b): in questo caso, dopo il milestone, avremmo avuto un intervallo di attesa lungo τ sul centro *A* (il tempo necessario all'arrivo dell'utensile β) ed uno lungo $\tau + \Delta$ sull'altra (il tempo necessario a trasportare l'utensile α da S_A a C_B).

Lo stesso discorso può ripetersi simmetricamente scambiando i ruoli dei due

centri, ossia supponendo che la risorsa critica sia stavolta la navetta *B*, il che dà luogo a due ulteriori casi possibili, rappresentati in Fig.2.16. Osserviamo che le attese a valle dei punti di disaccoppiamento sono, rispettivamente per i centri *A* e *B*, in un caso $2\tau+\Delta$ e 0, nell'altro $\tau+\Delta$ e τ. Adesso, il quadro è completo.

Va sottolineato che la situazione che si presenta nel nostro esempio (Fig.2.13) e quelle delle Figg.2.14 e 2.15 sono *mutuamente esclusive*, nel senso che partendo da un certo milestone (nodo del grafo) ci si può trovare in un caso, nell'altro o nell'altro ancora. Tuttavia, partendo da nodi diversi, si possono benissimo verificare tutti.

In definitiva, abbiamo che, se quattro operazioni possono dare luogo ad un conflitto doppio, esistono alpiù *quattro* possibili situazioni che tale conflitto può produrre, a valle del punto di disaccoppiamento. Vediamo di ricapitolarle, e di classificarle in base al ritardo che viene imposto *a valle* del punto di disaccoppiamento:

I) $2\tau+\Delta$ sul centro *A* e 0 sul centro *B* — Figg. 2.14(b) e 2.16(a);

II) $\tau+\Delta$ sul centro *A* e τ sul centro *B* — Figg. 2.13(b) e 2.16(b);

III) τ sul centro *A* e $\tau+\Delta$ sul centro *B* — Figg. 2.13(a) e 2.15(b);

IV) 0 sul centro *A* e $2\tau+\Delta$ sul centro *B* — Figg. 2.14(a) e 2.15(a).

Osserviamo che nei casi *I* e *IV* è compresa anche la situazione che si verifica a valle di un *conflitto semplice*.

In definitiva, la casistica relativa ai conflitti doppi può essere riassunta come indicato nella Tab.2.2, in cui è anche indicata la notazione usata in Fig.2.17 per indicare i nodi corrispondenti ai conflitti doppi. Infatti, volendo rappresentare sul grafo le alternative possibili per risolvere i vari conflitti, dovremo aggiungere *quattro* nodi e collegare il nodo predecessore ad essi con altrettanti archi, ciascuno pesato con la distanza tra i due milestone. Ancora una volta, a partire da ogni milestone, è possibile proseguire lo schedule normalmente fino a giungere ad un nuovo conflitto, o alla conclusione delle sequenze.

2.5.4.3 *Minimizzazione del tempo di completamento.*

Il grafo che si ottiene per il nostro esempio è raffigurato in Fig.2.17. Si noti che, a partire dal punto $B_3|A_2$, il conflitto tra le operazioni A_4 e B_5 è semplice; coerentemente con quanto osservato prima, il punto di disaccoppiamento relativo al caso in cui viene data la precedenza a B_5 è identico a quello considerato nel caso *I* del

conflitto doppio (Fig.2.14(b)), in cui il conflitto doppio è di fatto ricondotto a un conflitto semplice. Per questo l'arco è raccordato direttamente col nodo $B_4B_5|A_4A_5$.

Gap Temporale	**risorsa critica**	**modalità di scambio**	**nodo generato**
$\mid F(A_i) - F(B_{j-1}) \mid = \tau + \Delta$	robot (Fig.2.13)	il robot scambia gli utensili in $\boldsymbol{S_a}$ con la navetta $\boldsymbol{A}$	$\boldsymbol{B_{j-1}\mid_r A_{i+1}}$
		il robot scambia gli utensili in $\boldsymbol{S_b}$ con la navetta $\boldsymbol{B}$	$\boldsymbol{A_i\mid_r B_j}$
$\tau + \Delta = F(A_i) - F(B_{j-1}) = 2\tau + \Delta$	navetta A (Fig.2.15)	la navetta $\boldsymbol{A}$ scambia gli utensili in $\boldsymbol{S_a}$ con il robot	$\boldsymbol{A_i\mid_s B_j}$
		la navetta $\boldsymbol{A}$ scambia gli utensili in $\boldsymbol{C_a}$ con il centro	$\boldsymbol{B_{j-1}\mid_s A_{i+1}}$
$\tau + \Delta = F(B_{j-1}) - F(A_i) = 2\tau + \Delta$	navetta B (Fig.2.16)	la navetta $\boldsymbol{B}$ scambia gli utensili in $\boldsymbol{S_b}$ con il robot	$\boldsymbol{B_{j-1}\mid_s A_{i+1}}$
		la navetta $\boldsymbol{B}$ scambia gli utensili in $\boldsymbol{C_b}$ con il centro	$\boldsymbol{A_i\mid_s B_j}$

Tabella 2.2. Riepilogo dei possibili conflitti doppi.

Ovviamente, come già per *2JSP*, a ogni cammino da S a F sul grafo G corrisponde una sincronizzazione ammissibile del nostro problema: la lunghezza del cammino è proprio pari alla durata dello schedule, e di conseguenza risulta determinata anche la temporizzazione delle risorse del sistema di movimentazione – navette e robot.

; quindi abbiamo che nel caso peggiore il tempo necessario a risolvere un'istanza del problema P è $O(n^3)$.

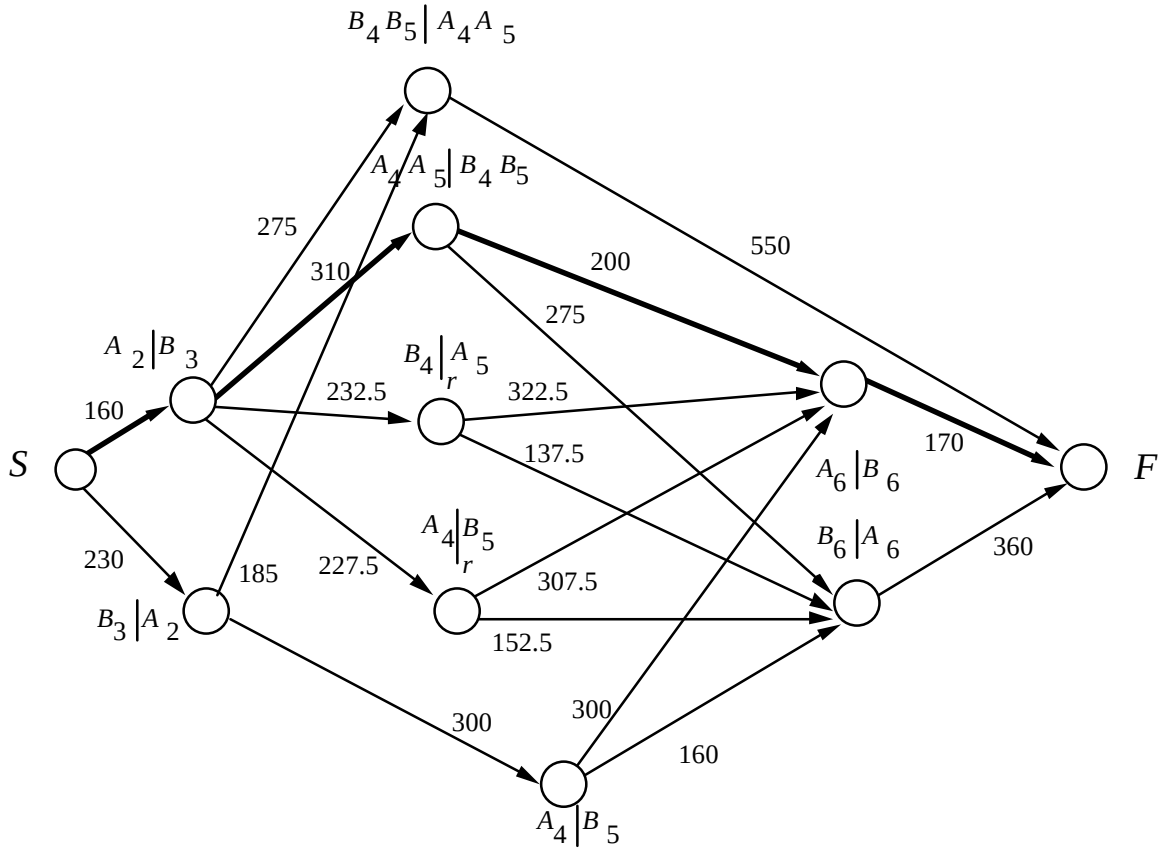


Figura 2.17. Il grafo relativo all'esempio.

2.5.5 Determinazione concorrente del parco utensili e della loro sincronizzazione

Nei paragrafi precedenti ci siamo occupati del problema di sincronizzare nel modo migliore possibile le operazioni dei due centri, sulla base della movimentazione degli utensili associata a queste operazioni.

Dunque, l'impatto sulla produttività della configurazione del magazzino utensili è legato al verificarsi, all'atto dell'esecuzione dei part program, di conflitti nell'uso degli utensili: ogni volta che uno stesso utensile – presente in copia unica – è richiesto da ambedue i centri di lavorazione, uno dei due è costretto ad attendere il rilascio dell'utensile da parte dell'altro. A livello di progetto, si può allora intervenire eliminando a priori la possibilità di conflitti, *duplicando* quegli utensili che appaiono maggiormente critici dal punto di vista della loro influenza sulla produttività, per i conflitti che determinano. In altre parole, mentre il problema *P* riguarda la sola sincronizzazione, data la disponibilità di certi utensili nel magazzino centrale, si può pensare di affrontare in modo *concorrente* (Lee e Mirchandani 1988) il problema complessivo dell'attrezzaggio (ovvero della composizione del magazzino centrale utensili) e della sincronizzazione degli utensili stessi, secondo un approccio in definitiva analogo a quello visto nel §2.4 per la cella con assegnamento "statico" di

utensili a macchine: in questo caso, ciò che non varia nel corso del normale funzionamento del sistema è la configurazione degli utensili nel magazzino centrale, che va decisa tenendo conto della movimentazione degli utensili da un centro all'altro. L'idea di base, dunque, è quella di porre nel magazzino utensili centrale due copie di quello o quegli utensili che risultassero più critici dal punto di vista della loro influenza sulla produttività. Tuttavia, valutare quanto un utensile sia critico non è, in generale, semplice. Come vedremo in seguito (§2.5.5.2), infatti, anche se la duplicazione di un certo utensile rimuove un conflitto che causava una lunga attesa su una delle due macchine, se ne possono presentare di nuovi, rendendo poco interessante la riduzione di T_c che si sperava di ottenere. D'altronde, è chiaro che la duplicazione di un gran numero di utensili porterà alla scomparsa di tutti i conflitti che hanno influenza sul valore del tempo di completamento, ma questo può essere inaccettabile dal punto di vista dei costi, in generale troppo elevati.

Il problema è dunque multiobiettivo: si vuole minimizzare il tempo di completamento dei due job, ma allo stesso tempo duplicare pochi utensili. In particolare, ci occuperemo del problema di determinare la sincronizzazione ottima, *avendo la possibilità di duplicare un numero di utensili non superiore a un intero h*. Per semplicità, nel seguito trascureremo i ritardi introdotti dalla presenza di robot e navette (ossia supporremo $\tau=\Delta=0$), in quanto la loro presenza non modifica la struttura concettuale del problema (a parte le complicazioni discusse nel §2.5.4), mentre ci consente di utilizzare la rappresentazione grafica del problema introdotta nel §2.3. Il problema di cui ci occupiamo in questo paragrafo è dunque il seguente:

Duplicazione di h utensili e sincronizzazione di due job in una cella flessibile (P_h)

Date

Due sequenze di operazioni $A=\{A_1, A_2,\ldots, A_n\}$ e $B=\{B_1, B_2,\ldots, B_m\}$ che devono essere eseguite dai due centri di lavorazione; per ogni operazione A_i (B_j), la durata $t(A_i)$ ($t(B_j)$) e l'utensile richiesto $u(A_i)$ ($u(B_j)$);

Determinare

Il numero di copie (1 o 2) di ciascun utensile presenti in magazzino centrale;

Un assegnamento di istanti d'inizio a ogni operazione su ogni centro;

tali che

il numero di utensili presenti in doppia copia non sia superiore a h

il tempo di completamento T_c delle operazioni delle due sequenze sia minimo

---

Come risulta dall'enunciato del problema, nel seguito ragioneremo in termini di *numero* di utensili che è consentito duplicare, esprimendo dunque in questo modo i vincoli di budget: ossia, supponiamo che gli utensili abbiano lo stesso costo. Questo

potrebbe ovviamente non essere vero; la discussione di come questo potrebbe influire sull'efficienza degli approcci risolutivi verrà svolta alla fine del §2.5.5.3. Nel seguito, faremo riferimento alla rappresentazione grafica del problema P_0 introdotta nel §2.3.2, e all'esempio numerico sviluppato in quel paragrafo.

Dal punto di vista grafico, cosa vuol dire duplicare un utensile? Se un utensile è presente in doppia copia nel magazzino centrale, due operazioni che lo richiedono potranno essere svolte in parallelo; dunque, duplicare un utensile vuol dire *eliminare alcune zone proibite dalla regione* Ψ. Il problema P_h è dunque quello di individuare l'insieme di zone proibite che è più conveniente rimuovere, nel rispetto del vincolo di budget.

2.5.5.1 *Casi facili e difficili*

Il problema P_h, nella sua generalità, è piuttosto complesso. Infatti, occorre notare che la decisione di duplicare un utensile porta in generale alla scomparsa di un *certo numero di zone proibite*, in parti diverse di Ψ, con effetti dunque non solo "locali". Precisamente, in generale vale il seguente risultato:

Teorema 2.4 – *Il problema P_h è NP–completo in senso forte.*
DIM. La dimostrazione è per riduzione da CLIQUE (Agnetis e Oriolo 1993).

D'altro canto, in molte situazioni di interesse pratico il problema si può risolvere ancora in modo ottimo, e in tempo polinomiale. Ciò che consente di tracciare un confine abbastanza chiaro tra i casi "facili" e "difficili" del problema P_h è il fatto che valga o meno una condizione analoga a quella (δ) vista a proposito del problema *PROC*(2,2).

***Condizione* γ.** *Ogni utensile è usato da alpiù una sola operazione da almeno uno dei due job.*

Si noti che tale condizione è meno restrittiva della condizione δ (tutte le operazioni di ciascun job richiedono utensili diversi). Il suo significato grafico è immediato: essa implica infatti che tutte le zone proibite associate a ciascun utensile sono disposte sulla stessa linea verticale o orizzontale della griglia. In questo paragrafo vogliamo mostrare che, se la condizione γ vale, il problema è risolubile in tempo polinomiale.

La differenza rispetto all'algoritmo risolutivo di *2JSP*, ovvero P_0, sta nel fatto che occorrerà risolvere – ancora tramite algoritmi di programmazione dinamica – non più un'istanza di percorso minimo, bensì di *percorso minimo pesato*, problema in generale NP-completo, ma in questo caso, come vedremo, polinomiale.

2.5.5.2 *La rete pesata R.*

Riprendendo l'esempio in Tab.2.3, supponiamo di procedere in parallelo sui due centri di lavorazione fino a incontrare il primo conflitto. Esso si ha sull'utensile β: se tale utensile fosse presente in doppia copia nel magazzino utensili, sarebbe possibile proseguire il processamento in parallelo sui due centri per un periodo di tempo più lungo: precisamente, il primo conflitto si presenterebbe sull'utensile ε. Possiamo rappresentare tale situazione sulla regione Ψ come indicato in Fig.2.18: dall'origine parte una coppia di archi, a ciascuno dei quali sono ora associati *due valori*: il primo (*distanza*) rappresenta, come prima, la lunghezza del tratto rettilineo necessario per raggiungere il vertice successivo, il secondo (*peso*) indica invece il *numero di zone proibite* che è stato necessario rimuovere per passare dall'origine al vertice successivo, ossia, in questo caso, il peso dei due archi sarà 1.

Come si vede dalla Fig.2.18, se decidessimo di duplicare *anche* ε, il successivo conflitto si avrebbe sull'utensile α (operazioni A_5 e B_5), e gli archi incidenti i vertici superiore sinistro e inferiore destro della zona (A_5,B_5) recherebbero le etichette (20,2) e (80,2) rispettivamente. Se poi fosse possibile duplicare *anche* α, si avrebbe un'unica diagonale da O a Q, senza tratti orizzontali (il fatto che Ψ sia quadrata dipende dal fatto che la somma dei tempi operazione è la stessa sui due centri ma non è ovviamente un fatto generale), e dunque pesata con (0,3).

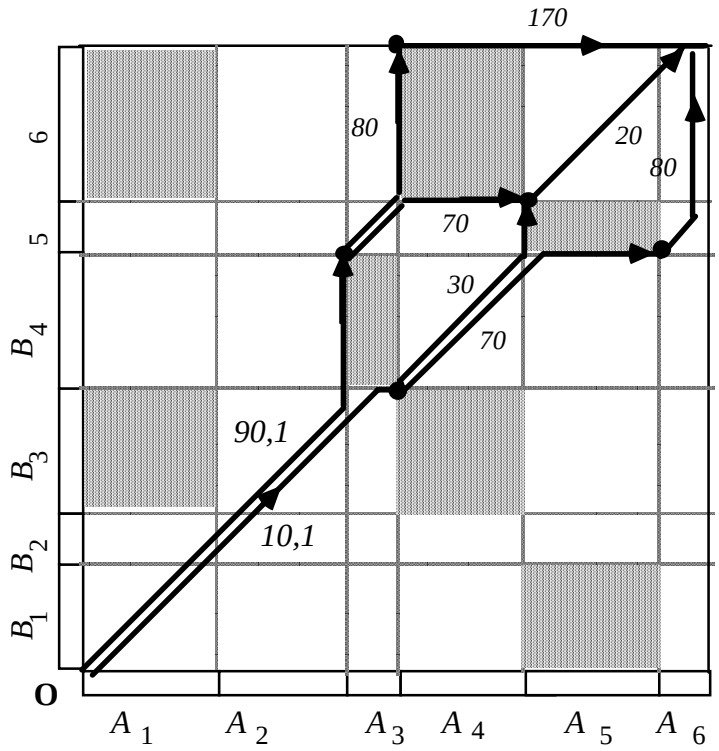


Figura 2.18. La regione Ψ dopo la duplicazione dell'utensile β.

La stessa costruzione può effettuarsi, chiaramente, a partire da ogni vertice di regioni proibite di Ψ, ottenendosi così in definitiva il grafo R di Fig.2.19, che per maggiore

chiarezza abbiamo rappresentato usando per i nomi dei nodi le stesse notazioni di quello di Fig.2.2. In esso, per *lunghezza* (*peso*) *di un cammino* intendiamo la somma delle lunghezze (pesi) degli archi del cammino.

A questo punto, consideriamo il seguente problema:

Rimozione di k zone proibite e sincronizzazione di due job (P_k*)

Dato
Il grafo R

Determinare
un percorso di lunghezza minima

tale che
il peso non sia superiore a k.

Prima di vedere come risolvere P_k*, vediamo quale relazione esiste tra esso e il problema P_h. A tale scopo, consideriamo un cammino da *O* a *F* su *R*, di peso *maggiore di zero*: ossia, consideriamo una soluzione ammissibile in cui è stata rimossa almeno una zona proibita della griglia. Ora, se la condizione γ è soddisfatta, tali zone proibite corrispondono necessariamente a utensili distinti, in quanto qualunque cammino ammissibile su Ψ, dopo aver lasciato una fascia orizzontale o verticale della griglia, non vi ripasserà più. Per questo motivo, dunque, il peso complessivo di un cammino da *O* a *F* su *R* rappresenta esattamente il numero di *utensili* che è stato necessario duplicare per avere la temporizzazione corrispondente.

Se la condizione γ non è soddisfatta, il numero di utensili duplicati lungo un cammino può risultare inferiore al peso del cammino, allorché uno stesso utensile contribuisce a determinare i pesi di due archi distinti: infatti, se uno stesso utensile è associato a più di una zona proibita, la sua duplicazione porta alla scomparsa di tutte queste zone proibite, e non solo della prima. Si consideri ad esempio il caso in Fig.2.20: mentre chiaramente la soluzione di P_2 è 0, dal grafo *R* si ha che la soluzione di P_2* è 20: siccome i pesi degli archi sono calcolati su base locale, non si riconosce che in realtà, la rimozione della zona proibita (A_1, B_1) porta alla scomparsa anche della zona (A_3, B_3), e dunque è possibile raggiungere *F* da *O* con un unico segmento diagonale.

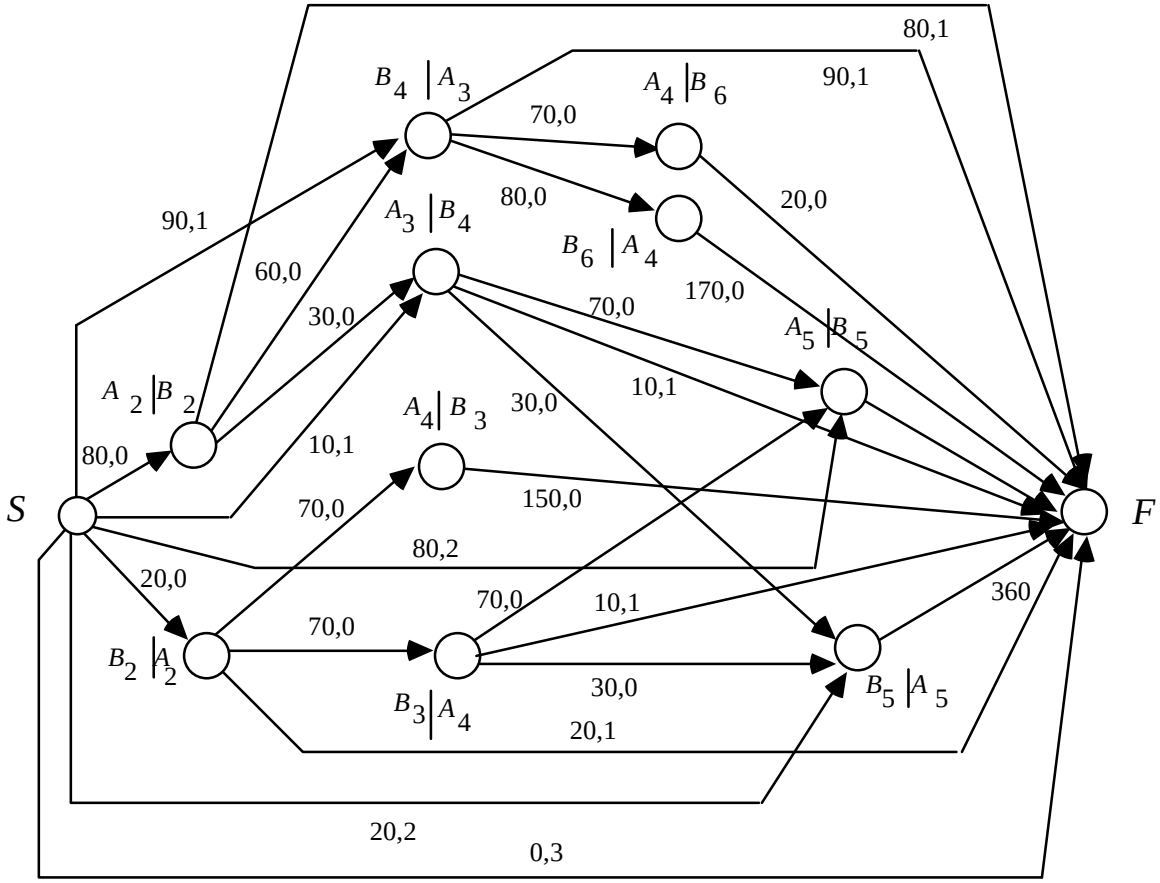


Figura 2.19. Il grafo R.

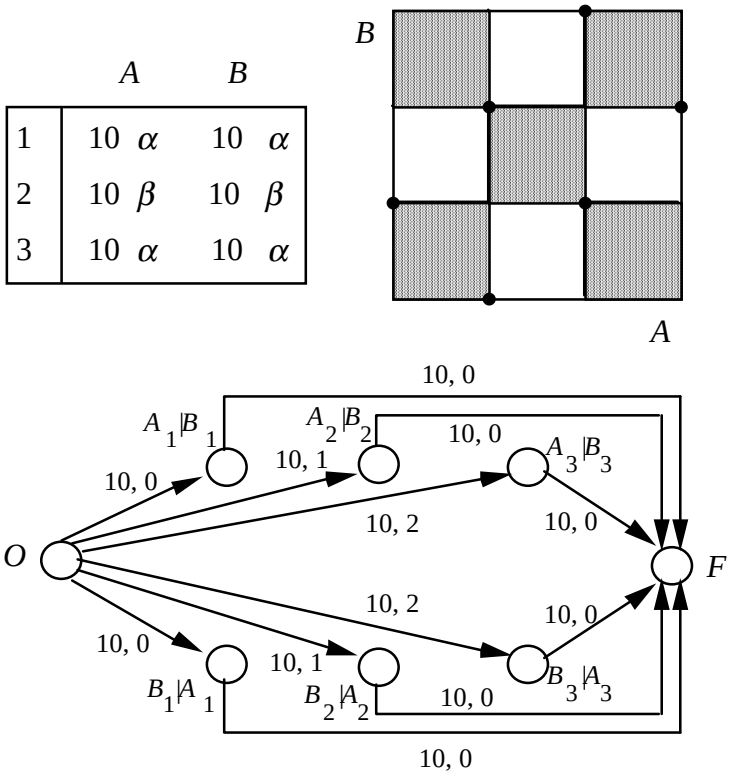


Figura 2.20. Esempio in cui la soluzione ottima di P_k* è diversa da quella di P_k.

Dunque, in generale la durata della sincronizzazione corrispondente alla soluzione

ottima di P_k* fornisce un *upper bound* al tempo che è realizzabile duplicando k utensili.

2.5.5.3 *Soluzione del problema P_k**

P_k* è evidentemente un problema di percorso minimo pesato, che in generale è NP-completo (si può facilmente ridurre da KNAPSACK). Nel nostro caso, però, osserviamo che il *numero di valori diversi* che può assumere il peso di un nodo è alpiù pari al numero di zone proibite che si possono incontrare tracciando una diagonale a 45º sulla griglia da O a F, e tale numero è ovviamente inferiore a $2n$, se con n indichiamo il numero di operazioni che ciascun centro deve effettuare. In questo caso, allora, il problema può risolversi con un algoritmo polinomiale, utilizzando ancora una volta il principio di ottimalità della programmazione dinamica.

Nel seguito, per semplicità notazionale indichiamo con lettere minuscole i nodi del grafo R (che corrispondono a vertici superiore sinistro o inferiore destro di zone proibite di Ψ). Sia $d(i,r)$ la *distanza* tra il nodo i e il nodo r, e w_{ir} il *peso* del relativo arco. Indichiamo con $F(r,h)$ la lunghezza del percorso minimo *di peso* = h dall'origine al nodo r. Consideriamo allora il generico nodo i che precede il nodo r in R. Poiché R è aciclico, è possibile numerare i suoi nodi in modo che ogni arco sia diretto da un nodo a indice inferiore verso uno a indice superiore; dunque sarà $i<r$. Il percorso minimo da O a r di peso = h passerà per uno dei predecessori di r, e dunque sarà dato dal valore della soluzione ottima fino al nodo i, più la distanza $d(i,r)$. Attenzione, però: la soluzione ottima da O a i deve essere di peso non superiore a $h–w_{ir}$, in quanto il peso dell'arco (i,r) viene poi aggiunto a tale soluzione e il tutto non deve eccedere h. In definitiva, il problema è risolto facendo uso della seguente equazione funzionale:

$$F(r,h) = \min_{i<r} \{ F(i,h–w_{ir}) + d(i,r) \} \qquad (8)$$

Chiaramente, la soluzione ottima del problema P_k* sarà data da $F(n,k)$.

Vale la pena di sottolineare che la condizione γ è solo sufficiente: ad esempio, risolvendo P_k* per l'esempio del §2.3.2, si ottiene la soluzione ottima di P_k, anche se in quel caso la condizione γ non è verificata. Peraltro, si può facilmente concepire un algoritmo euristico per P_k basato sulla risoluzione di P_k*. Prove sperimentali hanno mostrato, a questo proposito, che tale euristica trova l'ottimo in una grande quantità di casi, e in modo molto efficiente, anche grazie all'uso dell'algoritmo A* nella risoluzione del problema di percorso minimo. C'è infatti da osservare che in molti casi pratici, a meno che il numero di utensili diversi richiesti dalle varie operazioni

non sia molto piccolo rispetto al numero di operazioni, la probabilità che su un cammino da *O* a *Q* capitino due zone proibite causate da conflitti sullo stesso utensile, sono piuttosto basse (per dettagli si veda (Agnetis e Oriolo 1993)).

La (8) continua a essere valida anche se noi associassimo un *costo* diverso da 1 ad ogni zona proibita (tipicamente, il costo dell'utensile corrispondente): in tal caso, w_{ir} rappresenterebbe la somma dei costi di tutte le zone proibite che è necessario rimuovere per andare da *i* a *r*, e $F(r,h)$ la soluzione ottima da *O* a *r* di costo non superiore a *h*. Stavolta si noti che, dovendo *h* assumere valori limitati superiormente da una funzione dei costi delle zone proibite, l'algoritmo non è più polinomiale, bensì pseudopolinomiale, come qualsiasi algoritmo risolutivo esatto per un generico problema di percorso minimo pesato.

3. ROUTING NEI SISTEMI FLESSIBILI PIPELINE

3.1 Introduzione

La movimentazione dei materiali è un aspetto del sistema produttivo di fondamentale importanza. Una classe molto importante di sistemi di movimentazione è quella delle *linee seriali*, indicate in letteratura con molti nomi, come *pipelines, flow shops* (nella terminologia della teoria dello scheduling) o *tandem* (nella terminologia delle reti di code). In questo tipo di sistemi, i pezzi sono trasportati da un sistema di movimentazione unidirezionale, ed esiste un ordinamento totale tra le unità operatrici (che in questo capitolo chiamiamo *macchine*), cosicché i pezzi possono muoversi solo dalla macchina M_j alla M_{j+1} (Figura 1.8).

In generale, la topologia seriale è preferita a strutture più generali allorché i pezzi richiedono caratteristiche particolari per il sistema di trasporto — come ad esempio quando ci sono da movimentare pezzi ingombranti o pesanti — o, anche, allo scopo di ridurre la complessità e il carico di lavoro del supervisore che deve coordinare il flusso dei materiali.

In particolare, nelle assiemature meccaniche il sistema di trasporto può costituire un aspetto critico del sistema quando i pezzi sono difficili da manipolare (per motivi di peso e/o volume); nelle assiemature di sistemi elettronici il sistema informativo può essere assai complesso, se deve tener traccia e controllare lo svolgimento di migliaia di operazioni di inserzione di componenti richieste da ogni singola unità (ad esempio, nella produzione di computer, radar etc.).

Tra i vantaggi della struttura pipeline rispetto a sistemi di trasporto caratterizzati da una maggiore flessibilità di instradamento (§1.5) vanno menzionati: costi di progettazione e di installazione inferiori (sono necessari dispositivi fisici meno sofisticati e meno costosi), semplicità di supervisione, impossibilità di dare luogo a fenomeni di stallo (*deadlock*) proprio perché l'accesso alle macchine per ogni pezzo è disciplinato da una logica strettamente sequenziale. Inoltre, per una vasta classe di applicazioni, la struttura del *process plan* "combacia" particolarmente bene con quella del sistema di trasporto pipeline, come illustrato nel seguito. I punti deboli della struttura pipeline sono chiaramente relativi alla mancanza di instradamenti alternativi, il che implica il blocco dell'intero sistema allorché si verifica un guasto (per un'ulteriore discussione della struttura pipeline, si veda ad esempio il recente testo di Askin e Standridge (1992)).

La struttura pipeline è molto comune in alcuni sistemi flessibili di assiematura, quando ogni unità del prodotto ha un *componente principale*, e tutti gli altri componenti o sottoassiemi sono montati *su di esso* in una data sequenza. Esempi tipici sono: nell'assiematura di un circuito stampato, quando i chip sono inseriti sulla piastra di supporto; nell'assiematura dei motori di automobili, quando molti componenti sono montati su un telaio che si muove lungo una linea seriale di stazioni di lavoro etc. In questi casi, spesso, il problema significativo, ai fini della produttività dell'impianto, è il sequenziamento delle operazioni che coinvolgono il componente principale. Le altre operazioni riguardano tipicamente la preparazione e il posizionamento dei componenti che andranno montati sul componente principale, e per questo motivo le indichiamo come *operazioni di predisposizione*. Molto spesso è possibile *aggregare* le operazioni di predisposizione in singoli nodi dell'albero di assiematura: l'unità operatrice cui quel nodo sarà assegnato eseguirà, in realtà, l'intero insieme di operazioni di predisposizione (come indicato in Figura 3.1). Nel §3.2, ci occuperemo del problema di *assegnamento di operazioni a macchine* dopo che agli alberi di assiematura viene applicato il procedimento di aggregazione accennato prima. Come detto, tale struttura è legata al fatto che esiste un componente principale il quale "induce" nell'albero di assiematura un *cammino principale* da una foglia alla radice. La struttura risultante è quella di un *pettine* (Figura 3.1*b*), costituito da un *manico* e vari *denti*. Il manico corrisponde a operazioni che riguardano il componente principale, mentre ogni dente a un'operazione (aggregata) di predisposizione, eseguibile da una singola macchina e che coinvolge uno o più componenti del prodotto.

Nel §3.3 affronteremo un altro aspetto della gestione dei flussi nei sistemi flessibili di produzione: in particolare, ci occuperemo di *selezione dei part type* e *sequenziamento* dei pezzi sulle macchine in un sistema pipeline. Il modello cui spesso faremo riferimento è quello classico del *flow shop scheduling* (French 1982), tuttavia la situazione che analizzeremo è notevolmente diversa da quella classica, in quanto prenderemo in considerazione un ambiente fortemente caratterizzato dalla limitata disponibilità di risorse per l'immagazzinamento temporaneo dei pezzi, ovvero i *buffer*.

Due obiettivi contrastanti nella gestione dei flussi sono la massimizzazione della produttività e la minimizzazione del work–in–process. Quanto maggiore è l'utilizzo delle macchine tanto maggiore è la produttività, ma tanto maggiore risulta anche il numero complessivo di unità in lavorazione, e, di conseguenza, il livello dei magazzini di semilavorati tra le macchine.

I modelli e gli algoritmi che illustreremo nel §3.3 possono pensarsi come strumenti di supporto alle decisioni nel problema del dimensionamento dei buffer in un sistema seriale di lavorazione: i risultati che, a fronte delle informazioni relative a un piano di produzione tipico giornaliero, si ottengono, possono essere confrontati con quelli di altre analisi, che assumono invece capacità illimitata dei buffer; in questo modo si può avere un'informazione quantitativa su quanto influisce la dimensione del buffer sulla produttività.

D'altro canto, in molti casi la limitatezza o l'assenza dei buffer può essere un vincolo inerente al particolare processo in considerazione. Ad esempio, in certi processi termici le unità debbono passare attraverso una serie di forni a diverse temperature, e l'attesa di un'unità tra un forno e l'altro potrebbe portare un raffreddamento indesiderato; o anche, in taluni casi le "macchine" sono in realtà stazioni di lavoro in cui lo spazio a disposizione per le unità in attesa è molto limitato.

In taluni casi, la limitata capacità dei buffer può porre dei problemi perfino in sede di decisioni su quali insiemi di pezzi è più conveniente avere in produzione nello stesso intervallo di tempo: ossia, può essere determinante basare le decisioni di selezione dei part type su aspetti legati all'uso delle risorse di immagazzinamento che deriverebbero dalla produzione congiunta di certi part type. Nel §3.4 vedremo un modello di PL sviluppato per un sistema pipeline con 2 macchine, che consente di selezionare i part type in modo da massimizzare la durata dell'intervallo di tempo in cui il sistema è in grado di funzionare al massimo della propria efficienza, compatibilmente con la limitatezza del buffer tra le due macchine. Inoltre, una volta formati i lotti di produzione con tale modello di PL, il sequenziamento può essere realizzato attraverso l'uso di una semplicissima regola, che garantisce l'ottimalità rispetto all'utilizzazione delle macchine.

3.2 Problemi di routing nei sistemi pipeline

3.2.1 Introduzione

In questo capitolo analizzeremo il problema del routing in un sistema flessibile di assiematura di tipo pipeline. Supporremo che ogni macchina sia in grado di eseguire qualunque operazione (macchine identiche), se dotata dell'opportuno utensile. L'approccio al problema dell'attrezzaggio è in questo capitolo diverso rispetto a quello che vedremo nel §4.4: mentre lì le scelte di instradamento saranno vincolate dalla scelta di un certo attrezzaggio, qui supponiamo di attrezzare le varie macchine (che però in realtà potrebbero essere stazioni di lavoro con operai) sulla base dell'assegnamento delle operazioni alle macchine, oppure di assegnare operazioni e utensili in modo contestuale. L'approccio dunque non è esattamente un approccio di

decomposizione, ma è più vicino alla filosofia di un approccio *concorrente* (Lee e Mirchandani 1988).

Il problema complessivo della gestione dei flussi riguarda, come sappiamo (§1.6), oltre al routing, anche il sequenziamento dei pezzi sulle macchine (scheduling), allorché unità di tipo diverso sono presenti nel sistema allo stesso tempo.

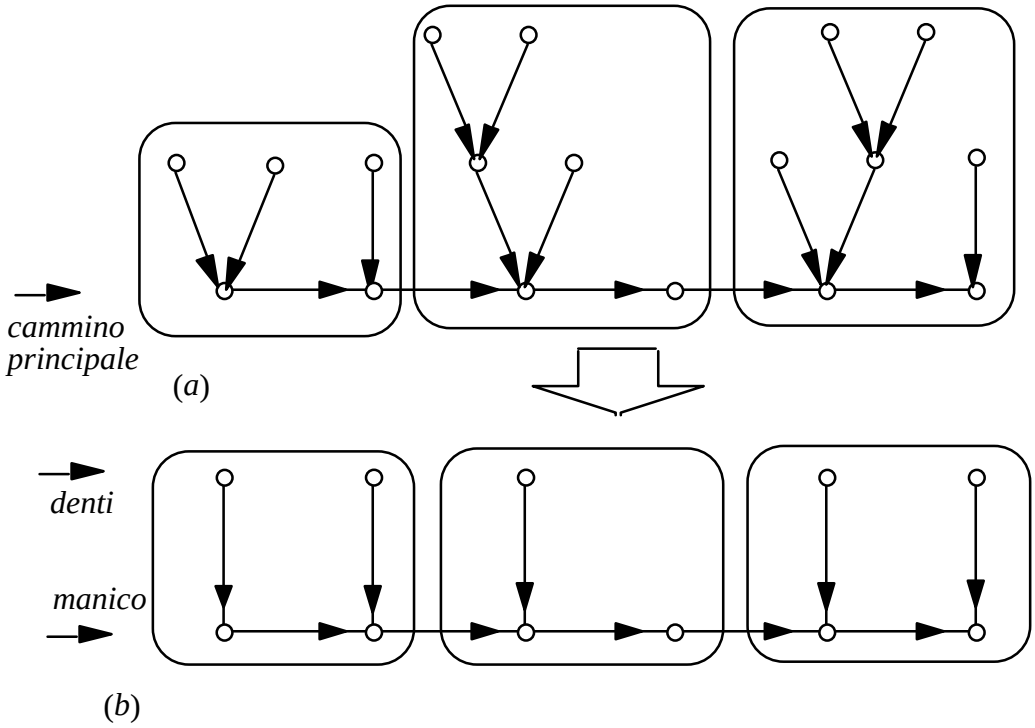


Figura 3.1.*a*) Albero di assiematura e cammino principale; *b*) La sua aggregazione (*pettine*), un possibile assegnamento di operazioni a macchine.

In questo capitolo ci occupiamo del problema del routing, rimandando al §3.3 per la trattazione di un particolare problema di scheduling di pezzi nei sistemi pipeline. Noi supporremo che la sequenza di unità di tipo diverso da produrre sia data, come in effetti accade in molte situazioni (ad esempio, quando la produzione è fortemente condizionata dalle date di consegna). Supporremo poi che l'ordine in cui vengono eseguite le operazioni di predisposizione è irrilevante, dal momento che ogni macchina può eseguire una sola operazione alla volta.

Al solito, l'obiettivo del routing è quello di massimizzare la produttività del sistema, o, equivalentemente, di massimizzare l'utilizzazione delle macchine. Tale obiettivo può poi tradursi in diverse funzioni–obiettivo, in genere dipendenti dal volume di produzione e dal mix.

Come già nelle celle flessibili (§2.1), anche nei sistemi pipeline l'assegnamento di operazioni a macchine che si vuole determinare è unico per tutti i pezzi dello stesso

tipo. Questo deriva soprattutto da motivi di semplicità gestionale e supervisiva, di economia nell'attrezzaggio dei centri, nonché dal fatto che — almeno storicamente — talora i centri di lavorazione sono costituiti da squadre di operai ciascuna delle quali dovrà eseguire sempre le stesse operazioni su ogni pezzo entrante nel sistema. Almeno fintantoché la produzione è omogenea, appare complicata e foriera di errori la strategia di fare eseguire, su tre pezzi consecutivi un certo insieme di operazioni, sui successivi due altre operazioni, e così via. Perciò, l'assegnamento di operazioni a macchine è unico e definisce una *partizione* dell'albero di assiematura.

Si noti che per i sistemi pipeline la successione di macchine che ogni unità dovrà visitare è prefissata ed è la stessa per tutte le unità; quello che è da determinare è invece l'assegnamento di operazioni alle macchine. Per quel che concerne i problemi di routing per sistemi pipeline, il modello più noto in letteratura è probabilmente il problema ASSEMBLY LINE BALANCING, che consiste nel cercare la partizione dei nodi dell'albero di assiematura (in realtà, si può trattare di un più generale grafo di assiematura) nel minimo numero di classi, col vincolo che la somma dei tempi dei nodi in ciascuna classe non ecceda un prefissato limite (*tempo di ciclo*). Per una panoramica su tale problema, si veda ad esempio il survey di Ghosh e Gagnon (1989). Un problema chiaramente correlato ad esso è quello che si ottiene scambiando l'obiettivo (numero di macchine) con il vincolo (tempo di ciclo), ossia si può cercare la partizione in un numero *fissato* di classi (ossia, di macchine), che minimizza il tempo di ciclo. Indicheremo tale problema con CTM (CYCLE TIME MINIMIZATION). Vedremo (§3.2.5) che tale problema nasce come sottoproblema della procedura risolutiva nel §3.2.6.

L'obiettivo collegato forse più direttamente alla massimizzazione della produttività è la *minimizzazione del tempo di completamento di tutte le unità*. Talora, questo obiettivo non può essere perseguito senza un'inaccettabile complessità degli algoritmi risolutivi: come vedremo anche nel §4, è necessario talvolta far ricorso a obiettivi "surrogati" quali il bilanciamento dei carichi di lavoro tra le macchine e la minimizzazione del costo complessivo di trasferimento dei pezzi. Invece, nel caso in esame vedremo un algoritmo che è in grado di fornire in tempo polinomiale una partizione dell'albero di assiematura tale da minimizzare il tempo di completamento. Tale procedura, come vedremo, fa uso di tecniche di programmazione dinamica.

3.2.2 Notazioni e formulazione dei problemi

Vediamo dapprima quali ipotesi particolari si applicano al sistema in questo capitolo.

— Ogni macchina può eseguire qualunque insieme di operazioni; tutte le macchine impiegano lo stesso tempo a eseguire una stessa operazione;
— Il sistema deve produrre un numero totale di q unità identiche. Il prodotto è caratterizzato da un albero di assiematura T, avente la struttura di pettine (§3.1). Un nodo del manico di T sarà indicato con h_i, e il dente adiacente (se esiste) con g_i. L'insieme S_i costituito da h_i e, se esiste, g_i è chiamato *sezione*. Sia n il numero delle operazioni di T. Il nodo h_i (g_i) è pesato con la durata τ_i (θ_i) della corrispondente operazione.

Al fine di mantenere la gestione delle operazioni al livello più basso possibile di complessità, introduciamo le ulteriori condizioni:

i) La macchina cui verrà assegnata un'operazione di predisposizione è la stessa che dovrà eseguire la corrispondente operazione di assiematura: ossia, se un dente g_i è assegnato a una macchina, anche il corrispondente nodo del manico h_i deve essere assegnato alla stessa macchina.

ii) Come già accennato nel §3.2.1, le operazioni assegnate ad ogni macchina sono eseguite da quella macchina su tutte le unità che entrano nel sistema. In altre parole, vogliamo determinare *un* instradamento, che verrà seguito da tutte le unità.

Nei prossimi paragrafi sarà discusso il seguente problema, *Problema di Routing Pipeline* (PROP), e verranno presentati algoritmi polinomiali per la sua soluzione:

***Problema di Routing Pipeline* (PROP)**

Dato
un sistema pipeline costituito da m macchine, che deve produrre q unità identiche e il loro albero di assiematura aggregato (pettine) T,

trovare
un assegnamento di operazioni a macchine

tale che
il tempo di completamento delle unità sia minimo.

Un *assegnamento di operazioni a macchine* è una m-partizione $\Pi + \{P_1, P_2, ..., P_m\}$ dei nodi del pettine T, ove le operazioni in P_j sono assegnate alla macchina M_j. Una partizione *ammissibile* di un pettine è tale che per ogni i, se $h_i \quad P_j$, allora anche $g_i \quad P_j$ (vedi Figura 3.1). Come discusso sopra, tutte le unità sono instradate secondo la stessa partizione del pettine.

Come vedremo meglio nel §3.2.7, PROP presenta notevoli analogie col problema CTM. La differenza tra PROP e CTM risiede nel diverso modo di realizzare il parallelismo tra le macchine. Per illustrare il concetto, vediamo un semplice esempio.

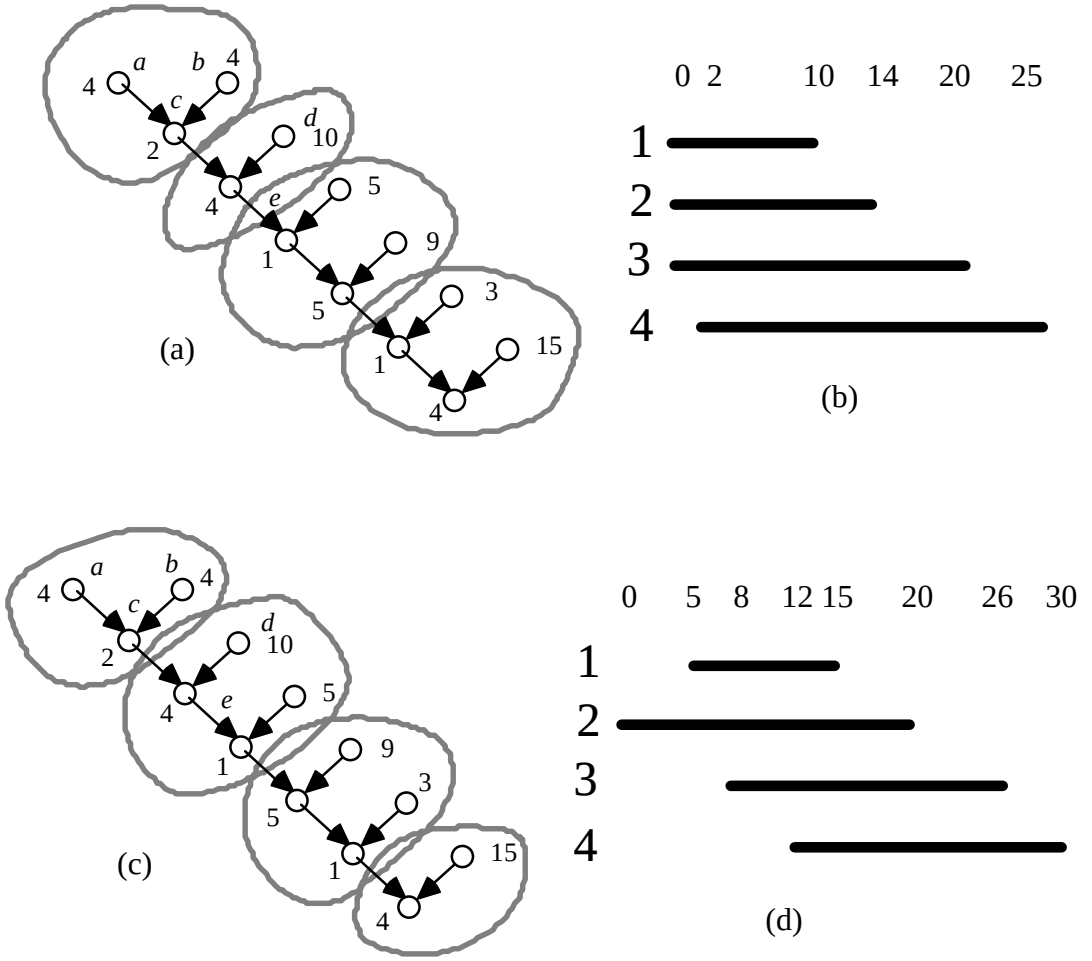


Figura 3.2. Albero di assiematura e due partizioni ammissibili.

Si abbia una singola unità ($q = 1$) che deve essere assiemata su un pipeline di $m = 4$ macchine. L'albero di assiematura aggregato è mostrato in Figura 3.2. Siano Π' e Π'' le due partizioni in Figg.3.2.*a* e 3.2.*c*, rispettivamente. Dal punto di vista del bilanciamento dei carichi di lavoro, la partizione Π'' è migliore, in quanto dà luogo a un tempo di ciclo di 20; cionondimeno, Π' consente di avere un tempo di completamento minore. La partizione Π' assegna le operazioni *a*, *b*, e *c* alla macchina M_1. Durante l'esecuzione di tali operazioni, M_2 può eseguire *d*, che corrisponde a un dente, e perciò non è soggetta ad alcun vincolo di precedenza. Perciò, M_1 e M_2 possono lavorare in parallelo sulla stessa unità, fintantoché le operazioni assegnate a M_2 non richiedono la disponibilità di un sottoassieme non ancora rilasciato da M_1. Questo è ciò che avviene per l'operazione *e*: affinché M_2 la possa eseguire, M_1 deve avere già concluso *c* — e dunque *a* e *b*. In conclusione, si ottengono i due diagrammi

temporali indicati in Figura 3.2*b* e 3.2*d*, corrispondenti a Π' e Π''. I tempi di completamento sono pari a 25 e 30 rispettivamente.

Questo semplice esempio mostra che, allorché si considera l'assiematura di una o poche unità, il bilanciamento dei carichi di lavoro tra le macchine può essere un obiettivo poco significativo. Il motivo è che, per piccoli valori di q, il miglior risultato si ottiene massimizzando il grado di parallelismo tra diverse operazioni *sulla stessa unità*, come appunto avviene per il problema PROP. Tuttavia, per q sufficientemente grande, le macchine possono sempre lavorare in parallelo su unità distinte, e la massimizzazione del grado di parallelismo può essere correttamente espressa dall'obiettivo di bilanciare i carichi di lavoro (come in CTM).

3.2.3. Partizioni ammissibili e tempo di completamento

In questo paragrafo vedremo che relazione intercorre tra una partizione ammissibile di T e il tempo di completamento dei pezzi. Successivamente, vedremo un algoritmo di programmazione dinamica che risolve il problema. Per semplicità espositiva, vedremo dapprima il problema di instradamento per un singolo pezzo (ossia, con q = 1), poi quando il numero di pezzi è grande (nel senso che verrà specificato meglio nel seguito), e infine, il problema per un generico numero q di pezzi, in cui si combinano gli algoritmi dei due casi precedenti. Successivamente vedremo meglio la relazione che intercorre tra PROP e CTM. Supporremo sempre che ciascuna macchina è potenzialmente in grado di eseguire qualunque operazione, non vi sono tempi di set-up, non vi sono limitazioni nei magazzini utensili.

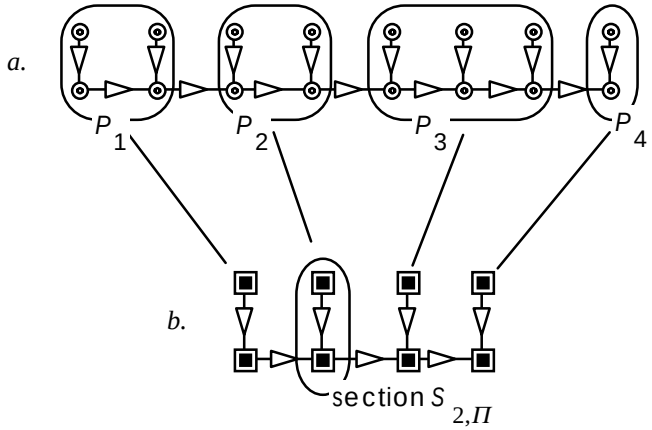


Figura 3.3. *a*) Partizione Π (m = 4) *b*) Il corrispondente pettine C_Π.

Data una partizione ammissibile $\Pi + \{P_1, ..., P_m\}$ di un pettine C, associamo a C e Π un *nuovo* pettine C_Π avente m sezioni, come segue (per semplicità, omettiamo l'indice Π dai nomi dei nodi): sia v_j il j-esimo nodo del manico di C_Π, e u_j il suo dente. La sezione $S_{j,\Pi}$ di C_Π corrisponde alla classe P_j di Π (Figura 3.3). Il peso di v_j (di u_j), indicato con $\tau_{j,\Pi}$ (con $\theta_{j,\Pi}$), è la *somma* dei pesi dei nodi h_i (dei nodi g_i) di P_j.

Si noti che, in base all'assegnamento Π, la somma $(\tau_{j,\Pi} + \theta_{j,\Pi})$ rappresenta il carico di lavoro per unità della macchina M_j .

Consideriamo ora q copie $C_\Pi^{[1]}$, $C_\Pi^{[2]}$,…, $C_\Pi^{[q]}$ di C_Π. Il j-esimo nodo del manico (il j-esimo dente) di $C_\Pi^{[s]}$ sarà indicato con $v_j^{[s]}$ ($u_j^{[s]}$). Definiamo ora un grafo G_Π nel seguente modo: G_Π contiene tutti i nodi e gli archi dei pettini $C_\Pi^{[1]}$, $C_\Pi^{[2]}$,…, $C_\Pi^{[q]}$; inoltre, c'è un arco da $v_j^{[s]}$ a $u_j^{[s+1]}$ per tutti gli $s = 1, 2, \ldots, (q-1)$ e per $j = 1, \ldots, m$; infine, c'è un nodo $v^{[0]}$ ed m archi orientati $(v^{[0]}, u_j^{[1]})$ per $j = 1, \ldots, m$. Gli archi $(v_j^{[s]}, u_j^{[s+1]})$ definiscono una relazione di precedenza tra operazioni: precisamente, obbligano la macchina M_j a completare dapprima le operazioni sull's-esima unità prima di iniziare a operare sulla $(s + 1)$-esima.

I nodi di G_Π sono pesati. Il peso di $v_j^{[s]}$ (di $u_j^{[s]}$) è pari al peso di v_j (di u_j) in C_Π. Il nodo $v^{[0]}$ ha peso nullo. Indichiamo con Φ un cammino da $v^{[0]}$ a $v_m^{[q]}$ su G_Π, e con $\mathrm{L}(\Phi)$ la sua lunghezza, ossia la somma dei pesi dei nodi di Φ. G_Π è mostrato in Figura 3.4: ogni nodo quadrato di G_Π corrisponde all'insieme di operazioni di predisposizione o di assiematura assegnate a una particolare macchina.

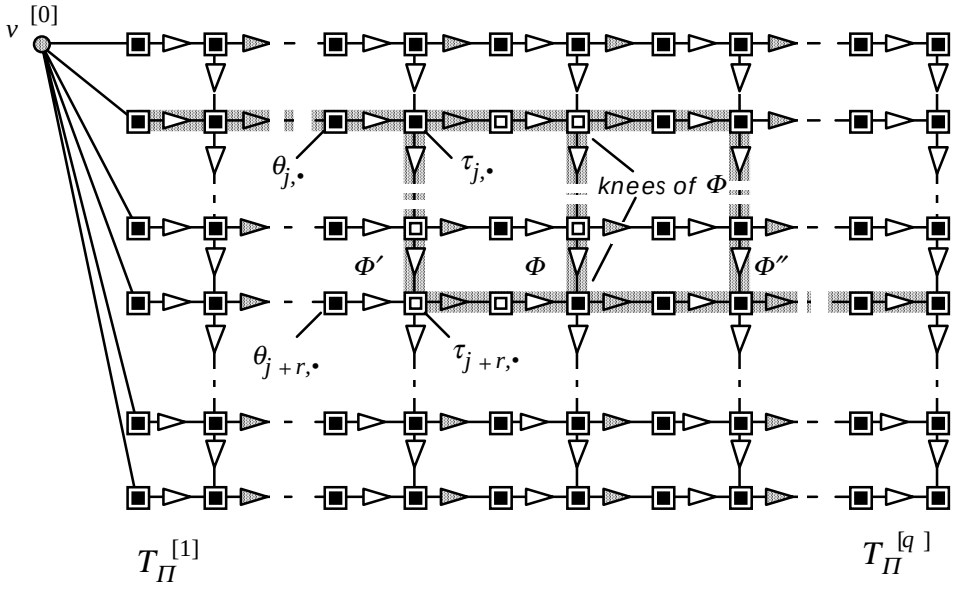


Figura 3.4. Il grafo G_Π e i cammini Φ, Φ' e Φ''. Φ e Φ' differiscono per i soli nodi bianchi.

Dato un assegnamento Π, esprimiamo ora il tempo di completamento come funzione delle durate delle operazioni:

Proposizione 1. *Dato un qualunque assegnamento* Π, *il tempo di completamento è pari alla lunghezza* $\mathrm{L}(\Phi)$ *del cammino critico* Φ *da* $\mathrm{v}^{[0]}$ *a* $\mathrm{v}_\mathrm{m}^{[q]}$ *sul grafo* G_Π.

Infatti, siccome la larghezza di G_Π (ossia il numero massimo di operazioni che possono essere svolta in parallelo) non eccede il numero m di macchine, G_Π può

vedersi come un grafo di attività (del tipo di quelli usati nella gestione progetti: PERT, CPM…) con risorse illimitate. Dunque, PROP può essere riformulato nel seguente modo:

Trovare una partizione ammissibile Π *di* C *tale che la lunghezza del percorso critico dal nodo* $v^{[0]}$ *al nodo* $v_m^{[q]}$ *sul grafo* G_Π *sia minima.*

Vediamo ora una proprietà dei percorsi critici su G_Π. Se un cammino Φ da $v^{[0]}$ a $v_m^{[q]}$ contiene i nodi $u_j^{[s]}$, $v_j^{[s]}$, $v_{j+1}^{[s]}$, per qualche $j \le m-1$, o i nodi $v_{j-1}^{[s]}$, $v_j^{[s]}$, $u_j^{[s+1]}$, per qualche $j \ge 2$, il nodo $v_j^{[s]}$ si chiama *ginocchio* di Φ (Figura 3.4). Chiaramente, ogni cammino Φ avente un ginocchio su $C_\Pi^{[s]}$, con $s < q$, ne ha un altro sullo stesso pettine. Vale allora il seguente teorema:

Teorema 1. *Tra i percorsi critici di* G_Π, *ne esiste almeno uno, sia esso* Ψ, *privo di ginocchia su* $C_\Pi^{[s]}$, *per* $s \ge 1$, $s \le q$.

DIM. — Supponiamo esista un cammino critico Φ che non verifica la suddetta condizione, cioè Φ ha un ginocchio su $C_\Pi^{[s]}$, con $s \ge 1$, $s \le q$, e si considerino i due cammini alternativi Φ' e Φ'' mostrati in Figura 3.4. Dal momento che Φ è un percorso critico, si ha $L(\Phi) \ge L(\Phi')$, cioè

$$\theta_{j,\Pi} + \tau_{j,\Pi} + \tau_{j+1,\Pi} + \dots + \tau_{j+r-1,\Pi} \quad \ge \quad \tau_{j+1,\Pi} + \dots + \tau_{j+r-1,\Pi} + \tau_{j+r,\Pi} + \theta_{j+r,\Pi}$$

vale a dire

$$\theta_{j,\Pi} + \tau_{j,\Pi} \quad \ge \quad \tau_{j+r,\Pi} + \theta_{j+r,\Pi}$$

D'altro canto, $L(\Phi) = L(\Phi'')$ implica

$$\theta_{j,\Pi} + \tau_{j,\Pi} \quad \le \quad \tau_{j+r,\Pi} + \theta_{j+r,\Pi}$$

e dunque Φ, Φ' e Φ'' sono tutti e tre critici. Nota che, se $v_j^{[h]}$, $v_{j+r}^{[h]}$ sono le due ginocchia di Φ' (di Φ''), abbiamo $h < s$ ($h > s$). In altre parole, le due ginocchia di Φ' (di Φ'') sono più vicine a $C_\Pi^{[1]}$ (a $C_\Pi^{[q]}$) di quanto lo sono le ginocchia di Φ. Ripetendo questo ragionamento, si può arrivare a mostrare l'esistenza di un cammino critico che ha ginocchia solo su $C_\Pi^{[1]}$ (rispettivamente, su $C_\Pi^{[q]}$), e segue dunque la tesi. Lo stesso può dirsi per il caso in cui Φ ha più di due ginocchia.

Q.E.D.

Sia Ψ un cammino che soddisfa le condizioni del Teorema 1. Se un indice s (s ? 1, s ? q) è tale che $u_j^{[s]}$ e $v_j^{[s]}$ appartengono a Ψ, allora per tutti gli s>1, i nodi $u_j^{[s]}$ e $v_j^{[s]}$ appartengono a Ψ. In altre parole possiamo derivare la seguente immediata conseguenza del Teorema 1 (Fig.3.5):

Corollario 1. *La lunghezza* $L(\Psi)$ *di un percorso critico è la somma di due termini (equazione* (1)*): il primo rappresenta la lunghezza di un cammino critico su un singolo pettine, il secondo corrisponde al carico di lavoro del collo di bottiglia.*

$$L(\Psi) = \max_{h=1,\dots,m} \{ \theta_{h,\Pi} + \sum_{j=h}^{m} \tau_{j,\Pi} \} + (q-1) \max_{j=1,\dots,m} \{ \tau_{j,\Pi} + \theta_{j,\Pi} \} \quad (1)$$

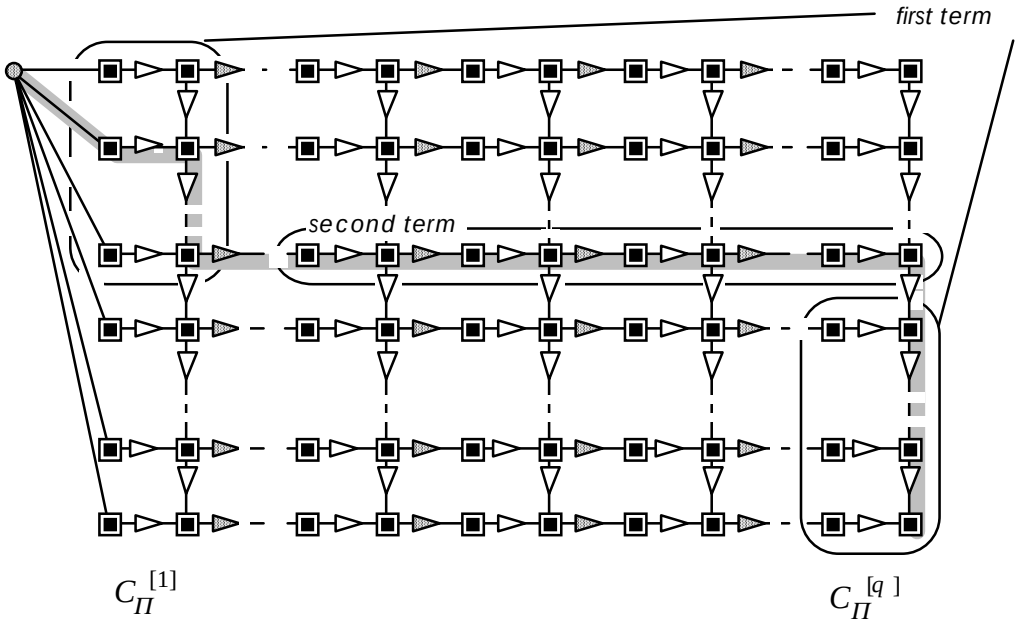


Figura 3.5. Il grafo G_{Π} e i termini dell'equazione funzionale (1). In grassetto il percorso critico.

Dall'equazione (1) dovrebbe essere intuitivamente chiaro che per valori sufficientemente grandi di q, minimizzare il carico di lavoro della macchina più carica e minimizzare il tempo di completamento sono obiettivi equivalenti: infatti il secondo termine nella (1) cresce all'aumentare di q, e per q sufficientemente grande domina l'altro.

Data una partizione Π, Si ha allora la seguente proposizione

Proposizione 2. *Se h^* e j^* sono gli indici per i quali si ottengono i massimi corrispondenti ai due addendi della (1), si ha $h^* \leq j^*$.*

DIM. — Supponiamo che sia $h^* > j^*$. Dunque, il percorso critico su C_{Π} partirebbe da una sezione a valle del collo di bottiglia. Ma poiché M_{j^*} è il collo di bottiglia, di certo si ha che $\theta_{j^*\Pi} + \tau_{j^*\Pi} \geq \theta_{h^*\Pi}$, e a maggior ragione $\theta_{j^*\Pi} + \sum_{i=j^*}^{h^*-1} \tau_{j^*\Pi} \geq \theta_{h^*\Pi}$. Quindi, se si considera il cammino su C_{Π} che parte da j^*, questo è almeno lungo quanto quello che parte da h^*, che dunque o non è critico oppure comunque non è unico.

Q.E.D.

Vedremo ora come applicare i risultati mostrati finora per sviluppare un algoritmo di programmazione dinamica per la soluzione di PROP nel caso di produzione omogenea. Inizialmente, considereremo i casi estremi $q = 1$ (§ 3.2.4) e q molto grande (§ 3.2.5): infatti, esiste un valore finito q^* tale che per $q = q^*$ la partizione ottima non cambia (§ 3.2.7). Nel paragrafo 3.2.6, vedremo come combinare i due metodi esposti per risolvere il caso generale.

Nel seguito, sia C_{hk} il sottografo di C indotto dalle sezioni da S_h a S_k. In particolare, C_{1k} (C_{kn}) è il sottografo indotto dalle prime k (le ultime $(n - k + 1)$)

sezioni. Se Π è una partizione ammissibile di C_{kn} in p classi, le sezioni di C_{kn} (rispettivamente, di C_{1k}) saranno assegnate alle macchine dalla M_{m-p+1} alla M_m (rispettivamente, dalla M_1 alla M_p). Definiamo ora il pettine $C_{kn}(\Pi)$ (il pettine $C_{1k}(\Pi)$) come quello ottenuto da C_{kn} (da C_{1k}) contraendo in una singola sezione le sezioni di C_{kn} (di C_{1k}) che appartengono alla stessa classe di Π.

3.2.4. Minimizzazione del tempo di completamento di un singolo pezzo

Consideriamo il caso $q = 1$. Il problema può essere ricorsivamente risolto sia considerando, al generico passo, il pettine C_{kn}, $k = n, \ldots, 1$ (programmazione dinamica all'indietro), o il pettine C_{1k}, $k = 1, \ldots, n$ (programmazione dinamica in avanti). I due metodi sono assolutamente equivalenti. Presentiamo qui quello all'indietro.

Indichiamo con $F_{kn}(p)$ la lunghezza minima di un percorso critico sul pettine $C_{kn}(\Pi)$, tra tutte le partizioni Π tali che $|\Pi| = p \le m$ e $1 \le k \le n$, cioè:

$$F_{kn}(p) = \min_{\Pi:\, |\Pi| = p} \{L(\phi_\Pi)\}$$

ove ϕ_Π è un cammino critico su $C_{kn}(\Pi)$.

Risolvere PROP vuol dire trovare $F_{1n}(m)$. Mostriamo ora come si calcola $F_{kn}(p)$. Supponiamo che, in una partizione ottima di C_{kn}, il pettine $C_{k,r-1}$ sia assegnato alla macchina M_{m-p+1}. Due casi sono possibili, a seconda che (*a*) la prima sezione di $C_{kn}(\Pi)$ appartenga al cammino critico oppure (*b*) no. Nel caso (*a*), la lunghezza $\mathrm{L}(\phi_\Pi)$ del cammino critico è data da:

$$\mathrm{L}(\phi_\Pi) = \sum_{i=k}^{r-1} \theta_i + \sum_{i=k}^{n} \tau_i$$

Nel caso (*b*), la lunghezza del cammino che contiene la prima sezione di $C_{kn}(\Pi)$ è dominata dal minimo tempo di completamento del pettine C_{rn} con $p - 1$ macchine. Per la Proposizione 1, questo tempo è pari a $F_{rn}(p - 1)$. Per calcolare $F_{kn}(p)$ dobbiamo scegliere il più piccolo tempo di completamento al variare di r. Quindi, l'equazione funzionale per $F_{kn}(p)$ sarà

$$F_{kn}(p) = \min_{k+1 = r = n} \left\{ \max \left\{ \sum_{i=k}^{r-1} \theta_i + \sum_{i=k}^{n} \tau_i \; ; \; F_{rn}(p-1) \right\} \right\} \qquad (2)$$

Si noti che l'equazione (2) è valida per $2 \le p \le \min\{m, n - k + 1\}$; infatti, p non può ovviamente superare il numero di sezioni del pettine C_{kn}. I valori iniziali di $F_{kn}(p)$ sono:

$$F_{kn}(1) = \sum_{i=k}^{n} (\tau_i + \theta_i) \qquad 1 = k = n \qquad (3)$$

Vediamo ora di valutare la complessità del calcolo di $F_{1n}(m)$. Un limite superiore è ovviamente dato da $O(mn^2)$. Infatti, il calcolo di $F_{kn}(p)$ richiede di

confrontare tra loro O(n) interi; d'altro canto, per calcolare $F_{1n}(m)$ dobbiamo conoscere gli O(mn) valori di $F_{kn}(p)$ ottenuti per $2 \le k \le n$ e $1 \le p \le m-1$.

Una procedura leggermente più sofisticata consente di migliorare questo valore di complessità. Per vedere ciò, si consideri, tra tutte le partizioni ottime di C_{kn} in p classi, una tale che alla macchina M_{m-p+1} sia assegnato $C_{k,r-1}$ con r massimo. Indichiamo una tale partizione ottima con $\Pi(p)$ e sia $r(p)$ tale massimo valore di r. Vediamo ora che $r(p)$ non può crescere al crescere di p:

Lemma 1. r(p + 1) = r(p).

DIM. — Sia Ψ_{p+1} il cammino contenente la prima sezione di $C_{kn}(\Pi(p+1))$. Siccome la lunghezza del percorso critico su $C_{kn}(\Pi(p+1)$ è data da $F_{kn}(p+1)$, risulta $L(\Psi_{p+1}) = F_{kn}(p+1)$. Siccome peraltro il tempo di completamento non può aumentare al crescere di p, si ha anche $F_{kn}(p+1) = F_{kn}(p)$, da cui

$$L(\Psi_{p+1}) = F_{kn}(p) \tag{4}$$

Supponiamo ora per assurdo che $r(p+1) > r(p)$ (Figura 3.6). Possiamo costruire allora una nuova partizione Π' di C_{kn} in p classi assegnando le sezioni dalla S_k alla $S_{r(p+1)-1}$ alla macchina M_{m-p+1}, e le altre sezioni di C_{kn} secondo la partizione ottima di $C_{r(p+1),n}$ in $(p-1)$ classi. Il tempo di completamento corrispondente alla nuova partizione Π' è dato da

$$\max\{L(\Psi_{p+1}), F_{r(p+1),n}(p-1)\} \tag{5}$$

Dalla definizione di $r(p)$ e dalla (2) possiamo scrivere $F_{kn}(p) = F_{r(p),n}(p-1)$. D'altra parte, siccome stiamo supponendo che $r(p+1) > r(p)$, dev'essere anche $C_{r(p),n} \ge C_{r(p+1),n}$, e quindi $F_{r(p),n}(p-1) = F_{r(p+1),n}(p-1)$. Dunque,

$$F_{r(p+1),n}(p-1) = F_{kn}(p) \tag{6}$$

Combinando (4), (5) e (6), possiamo concludere che Π' è ottima. Ma allora abbiamo costruito una partizione ottima avente un valore di r superiore a $r(p)$ e ciò contraddice l'ipotesi che $r(p)$ era il massimo valore di r in una partizione ottima di C_{kn} in p classi.

Q.E.D.

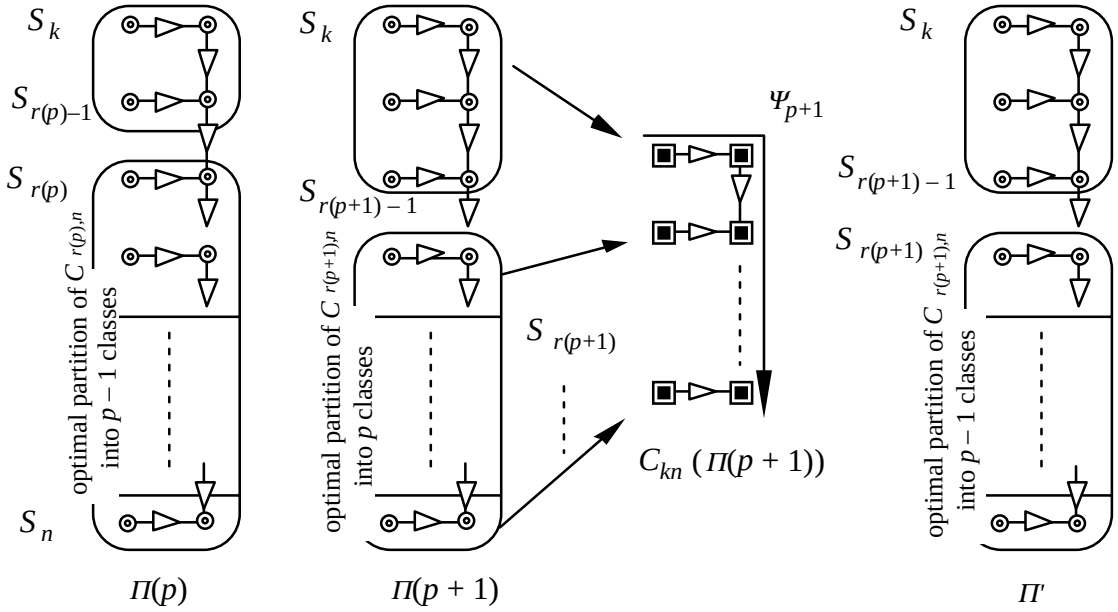


Figura 3.6. Dimostrazione del Lemma 1.

A questo punto, sulla base del Lemma 1, possiamo dimostrare che:

Teorema 2. $F_{1n}(m)$ *può essere calcolato in tempo* $O(n^2)$.

DIM.: Dal Lemma 1, possiamo limitarci a considerare l'insieme dei valori assunti da r nell'equazione (2) agli interi nell'intervallo $[k+1, r(p-1)]$. Se iniziamo la valutazione di $F_{kn}(p)$ in tale intervallo a partire da $r = r(p-1)$, al decrescere di r il primo termine della (2) decresce, mentre il secondo termine non decresce. Finché, per $r = r_0$, il secondo termine giunge a dominare il primo, e non è perciò più necessario diminuire ulteriormente r. Quindi, o $r(p) = r_0$, oppure $r(p) = r_0 + 1$. Al prossimo passo, il calcolo di $F_{kn}(p+1)$ inizierà da $r(p)$, e così via. In definitiva, per calcolare tutti i valori $F_{kn}(2)$, $F_{kn}(3)$, … *per un dato* k, ognuna delle sezioni da S_n a S_k viene scandita esattamente una volta, e dunque l'intero procedimento richiede un tempo $O(n)$. Poiché $1 \le k \le n$, segue la tesi.

Q.E.D.

Osserviamo qui che questo bound può essere ancora leggermente raffinato e portato a $O(nm \log n)$ se si impiega una ricerca binaria per trovare il punto di break-even r_0. Un risultato simile a quello espresso dal Teorema 2 può essere ottenuto relativamente a un algoritmo di programmazione dinamica in avanti, e segue lo stesso risultato di complessità.

3.2.5. Minimizzazione del tempo di ciclo

Supponiamo adesso $q = q^*$. Anche in questo caso svilupperemo una formula ricorsiva all'indietro. Come già osservato nel §3.2.3, si può intuire come in questo caso l'obiettivo di minimizzare il tempo di completamento si riduca a quello di

minimizzare il carico di lavoro della macchina più carica, ovvero di minimizzare il tempo di ciclo. Questo verrà dimostrato formalmente nel §3.2.7.

Data una generica partizione Π, il carico di lavoro $w(\Pi)$ è il più elevato tra i pesi delle sezioni di $C_{kn}(\Pi)$. La macchina cui viene assegnato quel carico di lavoro è detta *collo di bottiglia*. Sia $W_{kn}(p)$ il minimo valore assunto dal carico del collo di bottiglia al variare della partizione Π con $|\Pi| = p = m$ e $1 \le k \le n$, i.e.,

$$W_{kn}(p) = \min_{\Pi:\, |\Pi| = p} \{w(\Pi)\}$$

Per calcolare $W_{kn}(p)$ impiegheremo un metodo simile a quello descritto per il caso $q = 1$. Supponiamo che in una partizione ottima di C_{kn}, alla macchina M_{m-p+1} sia assegnato il sottopettine $C_{k,r-1}$. Sono possibili due casi, a seconda che (*a*) la prima sezione di $C_{kn}(\Pi)$ corrisponda al collo di bottiglia oppure (*b*) no. Nel caso (*a*), $w(\Pi)$ è dato da:

$$w(\Pi) = \sum_{i=k}^{r-1} (\theta_i + \tau_i)$$

Nel caso (*b*), il carico del collo di bottiglia $w(\Pi)$ è pari a $W_{rn}(p-1)$. Per calcolare $W_{kn}(p)$ scegliamo il più piccolo tra tutti i valori del carico del collo di bottiglia che si ottengono al variare di r. Quindi, l'equazione funzionale per $W_{kn}(p)$ è

$$W_{kn}(p) = \min_{k+1 = r = n} \{ \max \{ \sum_{i=k}^{r-1} (\theta_i + \tau_i) \; ; \; W_{rn}(p-1) \} \} \qquad (7)$$

Analogamente a quanto visto a proposito dell'equazione (2), anche la (7) vale per $2 \le p \le \min\{m, n-k+1\}$. I valori iniziali di $W_{kn}(p)$ sono:

$$W_{kn}(1) = \sum_{i=k}^{n} (\tau_i + \theta_i) \qquad 1 \le k \le n \qquad (8)$$

Per quanto concerne la complessità del calcolo di $W_{1n}(m)$, un upper bound semplice è dato da $O(mn^2)$. Come nel caso $q = 1$, si può provare con identico ragionamento che questo bound può essere migliorato, e portato a $O(\min\{n^2, mn \log(n)\})$.

Dal Corollario (1) sappiamo che $(q-1)W_{1n}(m)$ costituisce il "grosso" del tempo di completamento, quando $q = q^*$. Va aggiunto però che non tutte le partizioni tali che il carico di lavoro del collo di bottiglia è pari a $W_{1n}(m)$ sono ottime per PROP. Infatti, tra queste va scelta una per cui il primo addendo del secondo membro della (1) sia minimo. Una volta calcolato $W_{1n}(m)$, una partizione di questo tipo si ottiene semplicemente risolvendo un problema su un singolo pettine, modificando leggermente la formula ricorsiva (2) vista per $q = 1$, limitando il calcolo del secondo membro ai valori di r tali che

$$w_j = \sum_{i=k}^{r-1} (\theta_i + \tau_i) = W_{1n}(m).$$

3.2.6. Il caso generale

Passiamo ora al caso generale $1 < q < q^*$. L'approccio presentato per questo caso è basato sul risultato del Teorema 1 e utilizza in modo combinato gli algoritmi visti per i casi $q = 1$ e $q = q^*$.

Consideriamo un'istanza di PROP con q unità da produrre e un pettine C, e sia Π^q una partizione ottima. Si noti che in generale questa partizione non è unica. Siano W^q e F^q rispettivamente il carico del collo di bottiglia (i.e., il peso della classe più pesante di Π^q) e il tempo di completamento di una *singola unità* di prodotto (i.e., la lunghezza di un percorso critico su C_Π), che si hanno quando viene adottato l'assegnamento Π^q. Dalla (1), possiamo esprimere il tempo di completamento minimo di q unità come

$$T^q = F^q + (q - 1)\, W^q$$

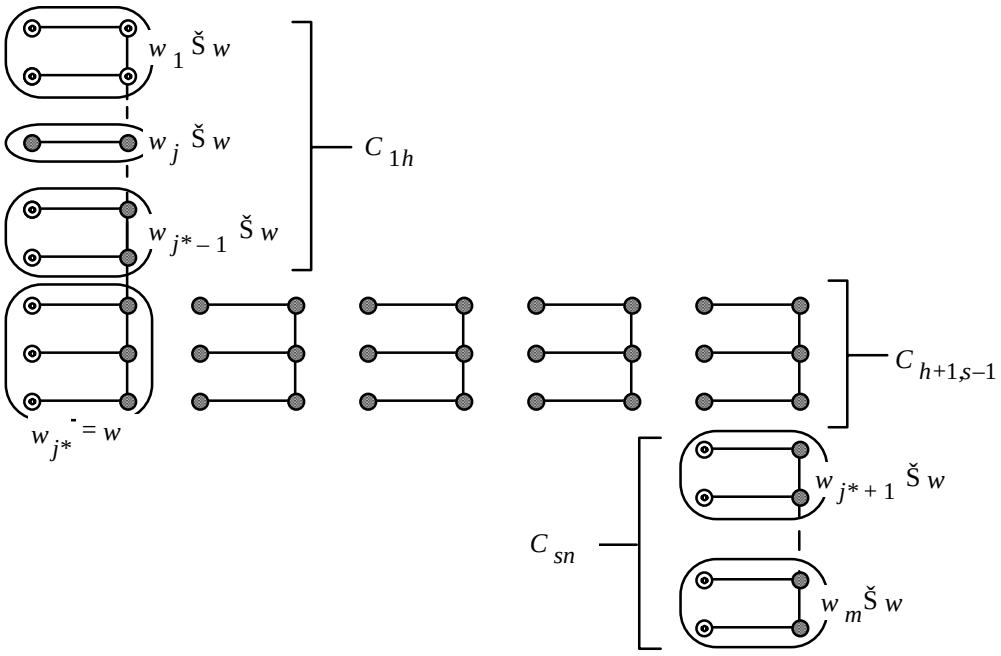


Figura 3.7. I nodi di G corrispondenti a un percorso critico su G_Π vincolato al fatto che il carico del collo di bottiglia non eccede W.

Se il valore W^q fosse noto a priori, si potrebbe facilmente ricondurre il calcolo del tempo di completamento minimo a quello già illustrato per il caso della macchina singola, con la sola differenza che ora il secondo membro della formula (2) va valutato solo limitatamente ai valori di r tali che

$$\sum_{i=k}^{r-1} (\theta_i + \tau_i) \;=\; W^q \qquad (9)$$

ovvero, possiamo scrivere che

$$F^q = \min \{\, F_{1n}(m) \mid w(\Pi) = W^q \}$$

(Si noti che questo controllo sul valore del secondo membro della (2) non altera la complessità computazionale dell'algoritmo). Tuttavia, il valore di W^q non è noto a priori, e dunque andrà calcolata la quantità $F(W) = \{F_{1n}(m) \mid w(\Pi) = W\}$ per tutti i possibili valori che W, carico di lavoro del collo di bottiglia, può assumere, e scegliere poi quello che dà luogo al minimo tempo di completamento.

Nella Fig.3.7 è raffigurata la situazione in cui $W = \sum_{i=h+1}^{s-1} (\tau_i + \theta_i)$. I nodi grigi individuano un percorso critico. Si ricorda (Proposizione 2) che la macchina in corrispondenza della quale il percorso critico inizia (nella figura, M_j) ha sempre indice minore o uguale a quello della macchina collo di bottiglia.

Il procedimento è riassunto in Figura 3.8.

Algoritmo PROP_SINGLE_TYPE
begin
calcola $W_{1n}(m)$; $T^* := +8$;
for W ammissibile **do**
begin
calcola $F(W)$ per mezzo delle equazioni (2) e (9);
sia $w(\Pi)$ il carico di lavoro del collo di bottiglia nella partizione ottima per il singolo pettine con carico di lavoro non superiore a W
if $F(W) + (q-1)\, w(\Pi) > T^*$ **then** $T^* := F(W) + (q-1)\, w(\Pi)$
end
end PROP_SINGLE_TYPE.

Figura 3.8. L'algoritmo risolutivo di PROP.

Per quel che concerne la complessità dell'algoritmo proposto, vale il seguente risultato:

Teorema 3. PROP *può essere risolto in tempo* $O(\min\{n^4, mn^3\log(n)\})$.

DIM. — Come già osservato, $W_{1n}(m)$ può calcolarsi in tempo $O(n^2)$. Per ogni valore di W, il calcolo di $F(W)$ può essere effettuato in tempo $O(\min\{n^2, mn \log(n)\})$, secondo quanto visto per $q = 1$. Inoltre, i valori ammissibili di W sono $O(n^2)$: infatti, essi sono limitati superiormente dal numero di possibilità diverse di scegliere i due estremi della classe da assegnare alla macchina collo di bottiglia. Segue quindi la tesi.

Q.E.D.

Vogliamo osservare qui che in pratica i valori di W da considerare nel ciclo dell'algoritmo risolutivo sono comunque molto pochi: infatti, non è necessario calcolare $F(W)$ per $W < W_{1n}(m)$: in questo caso, poiché $W_{1n}(m)$ è il minimo carico di lavoro del collo di bottiglia, non esisterebbe partizione ammissibile. Inoltre, non si ha

interesse a considerare valori di W più grandi del carico del collo di bottiglia di una partizione ottima con $q = 1$, come vedremo nel §3.2.7, Teorema 4. Si noti infine che la complessità dell'algoritmo non dipende dalla dimensione q del lotto.

L'approccio descritto consente di tenere facilmente in considerazione eventualità quali capacità limitata dei magazzini portautensili e/o limitazioni nello spazio di lavoro delle macchine. Queste restrizioni si traducono nel fatto che non qualunque set di utensili sarà assegnabile a ciascuna macchina. E' chiaro che in presenza di simili restrizioni PROP potrebbe perfino non ammettere soluzioni ammissibili.

3.2.7. Relazione tra PROP *e* CTM

Come abbiamo già avuto modo di osservare, l'equazione (1) implica che all'aumentare di q, minimizzazione del tempo di completamento e del carico del collo di bottiglia tendono a diventare obiettivi equivalenti. Vogliamo ora investigare meglio questo aspetto del problema.

Si ha il seguente teorema:

Teorema 4. *Data una qualunque sequenza di partizioni ottime* $\{\Pi^q \mid q = 1, 2, \dots\}$, *la sequenza* $\{W^q\}$ *è non crescente; la sequenza* $\{F^q\}$ *è non decrescente.*

DIM. — Ovviamente, si ha $F^1 \le F^2$, e, dalla (1), pure $F^1 + W^1 \ge F^2 + W^2$. Quindi, $W^1 \ge W^2$. In generale, per $q \ge 2$, abbiamo che Π^q è ottima per q unità ma non per $(q - 1)$ unità. Quindi:

$$F^{q-1} + (q-1)\, W^{q-1} \ge F^q + (q-1)\, W^q \tag{10}$$

d'altro canto, Π^{q-1} è ottima per $(q - 1)$ unità ma in generale non per q:

$$F^q + (q-2)\, W^q \ge F^{q-1} + (q-2)\, W^{q-1} \tag{11}$$

Sommando le disuguaglianze (10) e (11), otteniamo:

$$W^q \le W^{q-1} \tag{12}$$

Se ora moltiplichiamo ambo i membri della (12) per $(q - 2)$, e sommiamo il risultato alla (11), abbiamo

$$F^q \ge F^{q-1}$$

Q.E.D.

Sia ora $W^\infty = W_{1n}(m)$ (§ 3.2.5) il minimo valore del carico di lavoro per unità del collo di bottiglia, ossia il valore del minimo tempo di ciclo.

Teorema 5. *Esiste un valore finito* q^* *tale che, per ogni* $q \ge q^*$, $W^q = W^\infty$ *e quindi* Π^q *è ottima per* CTM.

DIM. — Come conseguenza del Teorema 4, $W^q \ge W^\infty$ per ogni valore di q. Mostriamo ora che c'è un valore di q per il quale $W^q = W^\infty$. Supponiamo infatti per assurdo che non esista. Consideriamo allora una partizione Π' tale che il suo carico del collo di bottiglia sia W^∞. Sia F' il tempo di completamento di una singola unità se si adotta la

partizione Π' . Dalla (1), il tempo di completamento di q unità è $F' + (q - 1)\, W^{\infty}$. Dalla definizione di F^q e W^q, si ha $(F^q - F') + (q - 1)(W^q - W^{\infty}) \leq 0$. Poiché abbiamo supposto $W^q > W^{\infty}$, abbiamo anche $q \leq (F' - F^q)/(W^q - W^{\infty}) + 1$, che è ovviamente falso per q sufficientemente grande.

Q.E.D.

Come già anticipato alla fine del §3.2.5, da questo teorema e dalla (1) si ha che per $q \geq q^*$, il tempo di completamento di una *singola* unità ottenuto quando si adotti una partizione $\Pi^{\infty} + \Pi^q$ (i.e., il valore F^{∞}) è quello minimo tra tutti quelli ottenuti con partizioni che hanno un carico di collo di bottiglia pari a W^{∞}. Ossia, per $q \geq q^*$, il tempo di completamento ottimo è dato da $F^{\infty} + (q - 1)\, W^{\infty}$.

Si noti che mentre Π^{∞} è ottima sia per CTM che per PROP, una partizione ottima per CTM può non esserlo per PROP, anche se $q \geq q^*$. E' interessante notare che, come W^{∞}, anche F^{∞} può essere facilmente calcolato a priori usando l'equazione (2) col vincolo (9). Infatti, F^{∞} è pari alla minima lunghezza di un percorso critico su un singolo pettine, con il vincolo che il peso di nessuna classe ecceda il valore W^{∞}. Dal Teorema 4 sappiamo che $F^{\infty} \geq F^1$ e che, per qualsiasi partizione ottima Π^1, $W^{\infty} \leq W^1$.

Si osservi che se $F^1 = F^{\infty}$, Π^{∞} è ottima per ogni $q \geq 1$, ossia $q^* = 1$. D'altro canto, se $F^1 < F^{\infty}$, allora $q^* \geq 2$. E' possibile derivare facilmente un lower bound e un upper bound al valore di q^* quando $q^* \geq 2$. Data una partizione ottima Π^1, poiché Π^{∞} è ottima per $q \geq q^*$, possiamo scrivere:

$$F^{\infty} + (q^* - 1)\, W^{\infty} \quad \leq \quad F^1 + \ (q^* - 1)\, W^1$$

che dà

$$q^* \ \geq (F^{\infty} - F^1\,) \,/\, (W^1 - W^{\infty})' + 1 \qquad (13)$$

D'altra parte,

$$F^{\infty} + (q^* - 2)\, W^{\infty} \quad > \quad F^{q^*-1} + \ (q^* - 2)\, W^{q^*-1}$$

Quindi, ricordando che $F^{q^*-1} \geq F^1$ e che, poiché si ha a che fare con numeri interi, $W^{q^*-1} - W^{\infty} \geq 1$, si ha

$$q^* \ < \ \ (F^{\infty} - F^{q^*-1}) \,/\, (W^{q^*-1} - W^{\infty})' + 2 < F^{\infty} - F^1 + 2 \qquad (14)$$

Ambedue questi bound possono essere facilmente calcolati a priori.

3.2.8. Un esempio

Vediamo di illustrare l'algoritmo presentato nel §3.2.6 con un esempio numerico. Si consideri il pettine in Figura 3.9, in cui sono indicate le durate delle operazioni, τ_i e θ_i. La linea di assiematura consiste di $m = 4$ macchine. Il pettine ha $n = 7$ sezioni. Vogliamo risolvere PROP per diversi valori di q.

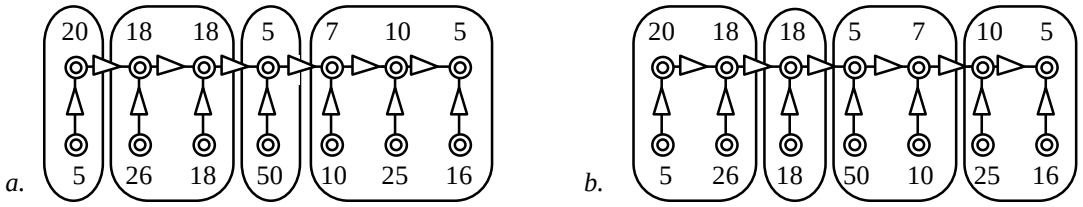


Figura 3.9. Pettine dell'esempio. *a*) Partizione Π^1; *b*) Partizione Π^∞.

Supponiamo dapprima $q = 1$. Applichiamo quindi le equazioni (2), (3). Partendo dalla fine del pettine, si calcolano i seguenti valori in modo immediato: $F_{77}(1) = 21$, $F_{67}(1) = 56$, $F_{67}(2) = 40$. Veniamo quindi ai valori $F_{57}(p)$: $F_{57}(1) = 73$, $F_{57}(2) = \min\{\max(32, 56); \max(57,21)\} = 56$ (la partizione corrispondente è $\Pi_{57}(2) = \{\{5\},\{6,7\}\}$), e infine $F_{57}(3) = \max(32, F_{67}(2)) = 40$.

Consideriamo ora i valori $F_{47}(p)$. In modo immediato si ha $F_{47}(1) = 128$. Seguendo la procedura descritta nel § 3.2.4, per calcolare $F_{47}(2)$ iniziamo dal fondo del pettine, e otteniamo:

$$F_{47}(2) = \min\{ \max(112, F_{77}(1)); \max(87, F_{67}(1)); \max(77, F_{57}(1)) \} = 77$$

Si noti che la prima classe di $\Pi_{47}(2)$ comprende solo la sezione S_4, cioè, in altri termini, il punto di break-even è $r(2) = 5$. Poiché, per il Lemma 1, $r(3) = r(2)$, e ovviamente $r(3) = 5$, possiamo immediatamente scrivere $F_{47}(3) = 77$. Si osservi che, se siamo interessati solo al valore finale $F_{17}(4)$, *non* è necessario calcolare $F_{47}(4)$, $F_{37}(4)$ e $F_{27}(4)$, poiché questi valori non compaiono nell'espressione usata per calcolare $F_{17}(4)$. Per lo stesso motivo non siamo interessati a $F_{37}(1)$, $F_{27}(1)$, $F_{27}(2)$, $F_{17}(1)$, $F_{17}(2)$, $F_{17}(3)$. inoltre, nel calcolo di $F_{37}(2)$ possiamo partire da $r = 7$, e in quello di $F_{37}(3)$ da $r = 6$. Vediamo in dettaglio i calcoli ulteriori, sottolineando il massimo tra due quantità, in modo da evidenziare il punto di break-even:

$$F_{37}(2) = \min\{ \max(\underline{148}, 21); \max(\underline{123}, 56); \max(\underline{113}, 73); \max(63, \underline{128})\} = 113$$

$$F_{37}(3) = \min\{ \max(\underline{113}, 56); \max(63, \underline{77})\} = 77$$

$$F_{27}(3) = \min\{ \max(\underline{167}, 40); \max(\underline{157}, 56); \max(\underline{107}, 77); \max(89, \underline{113})\} = 107$$

Siamo ora in grado di calcolare $F_{17}(4)$ (possiamo partire da $r = 5$).

$$F_{17}(4) = \min\{ \max(\underline{182}, 40); \max(\underline{132}, 77); \max(\underline{114}, 77); \max(88, \underline{107})\} = 107.$$

La partizione finale è perciò

$$\Pi^1 + \{\{S_1\}, \{S_2, S_3\}, \{S_4\}, \{S_5, S_6, S_7\}\} + \{C_{11}, C_{23}, C_{44}, C_{57}\}$$

(vedi Figura 19*a*), e dunque $F^1 = 107$. Questa partizione è ottima ma non è unica (ad esempio, un'altra partizione ottima è $\{C_{11}, C_{23}, C_{45}, C_{67}\}$); si può comunque osservare che $W^1 = 80$ in tutte le partizioni ottime.

Prima di passare al caso $q = 2$, vediamo il caso $q = 8$. L'algoritmo di § 3.2.5 è applicato in modo assolutamente simile a quanto visto per il calcolo di F^1. Si può vedere che in questo caso la partizione che minimizza il carico del collo di bottiglia è unica ed è data da

$$\Pi^{\infty} + \{\{S_1, S_2\}, \{S_3\}, \{S_4, S_5\}, \{S_6, S_7\}\} + \{C_{12}, C_{33}, C_{45}, C_{67}\}$$

(Figura 3.9*b*), per la quale $W^{\infty} = 72$ sulla macchina M_3. Con questa partizione si ottiene $F^{\infty} = 114$. Poiché $F^{\infty} > F^1$, possiamo applicare la disuguaglianza (11) e ottenere il bound

$$q^* = 1 + \cup (114 - 107) / (80 - 72)' = 2.$$

Supponiamo ora $q = 2$. Secondo l'algoritmo PROP_SINGLE_TYPE, dobbiamo trovare una partizione ottima per ogni valore ammissibile del carico del collo di bottiglia W. Benché W possa in linea di principio assumere $O(n^2)$ valori distinti, abbiamo già avuto modo di osservare che quelli che si ha interesse a considerare possono essere molti di meno. In questo caso, poiché $W^{\infty} = 72$, non esiste partizione ammissibile con $W < 72$. D'altra parte, essendo $W^1 = 80$, non è necessario considerare $W > 80$, dal momento che non si otterrebbe mai, in questo caso, un percorso critico sulla singola unità inferiore a $F^1 = 107$. Dunque, basta considerare i valori di W nel range [72, 80]. Generando tutte le classi ammissibili, vediamo che solo *tre* valori di W cadono in questo intervallo, e precisamente: $W = 72$ (peso di C_{45}), $W = 73$ (peso di C_{57}) e $W = 80$ (peso di C_{23}). Nel seguito, $T^q(W)$ indicherà il tempo di completamento vincolato al fatto che il collo di bottiglia ha un carico di lavoro pari a W.

Utilizzando le espressioni (2) e (9) si può risolvere dunque il solito problema di partizione del singolo pettine, per i tre valori suddetti. Nel caso $W = 72$ (Fig.3.10(a)), tale valore viene effettivamente ottenuto con il sottopettine C_{45} assegnato alla macchina M_3 (ossia $W = w(\Pi)$), e il corrispondente tempo di completamento vale $F(72) = 114$, e dunque, ricordando che $T^q(W) = F(W) + (q - 1)\, w(\Pi)$, nel nostro caso si ha $T^2(72) = 114 + 72 = 186$. Analogamente, nel caso $W = 73$ (Fig.3.10(b)), si ha che $F(73) = 114$ e quindi $T^2(72) = 114 + 73 = 187$. Infine, nel caso $W = 80$ (Fig.3.10(c)), si ha che $F(80) = 107$ e quindi $T^2(80) = 107 + 80 = 187$.

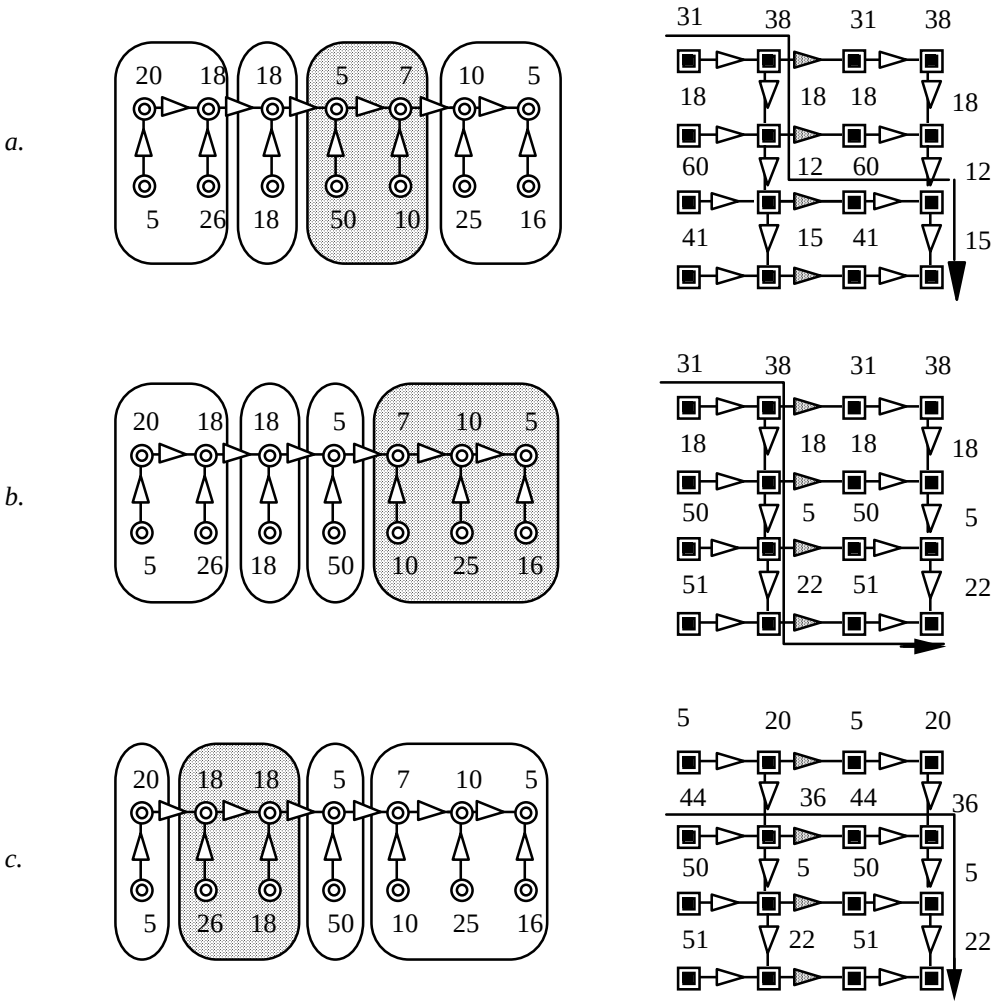


Figura 3.10. Partizioni ottime per $W = 72$, $W = 73$, e $W = 80$.

In definitiva, confrontando i tre diversi valori ottenuti per $T^2(W)$, possiamo concludere che la partizione ottima Π^2 per $q = 2$ è $P_1 = \{S_1, S_2\}$, $P_2 = \{S_3\}$, $P_3 = \{S_4, S_5\}$, $P_4 = \{S_6, S_7\}$, ottenuta con $W = 72$, per cui il tempo di completamento T^2 è pari a 186. Poiché $\Pi^2 + \Pi^\infty$, abbiamo $q^* = 2$, e il lower bound (11) vale con il segno di uguaglianza.

3.3. Scheduling nei sistemi pipeline in assenza di buffer

3.3.1 Introduzione

In questo capitolo affrontiamo un altro aspetto della gestione dei flussi nei sistemi pipeline: in particolare, ci occuperemo di *sequenziamento* dei pezzi sulle macchine in assenza di buffer in grado di ospitare i semilavorati in attesa di subire la lavorazione successiva. Di conseguenza, il modello che presenteremo corrisponde a un problema di *scheduling senza attese*, ovvero caratterizzato dal vincolo di *no–wait*.

Va sottolineato che l'interesse nei confronti dei problemi di sequenziamento senza attese è particolarmente sentito in concomitanza con la diffusione dei principi e delle metodologie produttive giapponesi: secondo tali principi, infatti, tutto il tempo che un pezzo trascorre nel sistema senza che su di esso vengano effettuate operazioni, è una perdita, in quanto un certo ammontare di capitale è stato investito in quel semilavorato, ed esso rimane improduttivamente ad attendere nei magazzini. Dunque, mentre un tipico approccio della gestione dei flussi è quello che corrisponde a massimizzare l'utilizzo delle macchine, l'approccio *just–in–time* pone invece l'enfasi sul prodotto, e cerca di organizzare la produzione orientandola alle esigenze del prodotto, anche se queste talvolta comportano un livello di utilizzazione delle macchine inferiore a quello teoricamente conseguibile.

In realtà, la filosofia just–in–time è ben più vasta e profonda e non può ridursi smplicemente a un metodo per il controllo dei magazzini. Ad esempio, un aspetto fondamentale nella realizzazione di un sistema just–in–time è legato all'abbassamento drastico dei tempi di *set–up*: per poter inseguire la variabilità della domanda produttiva nel modo migliore, è necessario avere la possibilità di cambiare rapidamente il mix produttivo, arrivando a produrre lotti consistenti, teoricamente, di un solo pezzo: ciò è possibile, evidentemente, solo se il tempo necessario per passare da un tipo di lavorazione ad un altro è sufficientemente basso. Per una discussione approfondita delle metodologie produttive just–in–time possono consultarsi vari testi, tra cui ad esempio quello di Chase e Aquilano (1989).

L'obiettivo dell'analisi riportata in questo paragrafo è stato quello di progettare algoritmi in grado di dare una valutazione quantitativa dell'utilizzazione massima (tempo di completamento minimo) che si può ottenere, in corrispondenza a uno scenario in cui ogni unità che entra nel sistema è vincolata a restarvi solo il tempo strettamente necessario alla sua lavorazione. Tali algoritmi dovevano essere *efficienti*,

cioè essere eseguibili su un calcolatore anche con potenze di calcolo limitate, e, possibilmente, in grado di garantire un certo errore massimo nel caso peggiore.

Come vedremo (§3.3.2), il problema della minimizzazione del tempo di completamento di un insieme di q lavori su m macchine seriali, con il vincolo di no–wait, è in generale molto difficile. Tuttavia il fatto che, in pratica, i pezzi da lavorare non siano tutti diversi, bensì appartengano solo a $b<<q$ lotti distinti, può essere utilmente sfruttato per aumentare l'efficienza degli algoritmi risolutivi. Questa situazione è simile a quella che vedremo nel §4.4, allorché l'ipotesi di lotti di grandi dimensioni consentirà di risolvere in modo estremamente efficiente il problema di routing.

In questo capitolo analizzeremo il problema in relazione al contesto di *sequenziare* un insieme di unità – tutte disponibili fin dall'inizio – su di un sistema con m macchine in serie, privo di buffer. Questo problema ricade esatamente nel formato di un modello di scheduling classico, noto con il nome di NO-WAIT FLOW SHOP. Nel §3.3.2 rivedremo alcuni concetti relativi a questo problema, nel §3.3.3 presenteremo alcuni risultati preliminari che servono allorché il problema viene visto nell'ambito di un sistema che deve lavorare diversi part type, mentre nel §3.3.4 verrà presentato l'algoritmo e, nei paragrafi successivi, verranno brevemente discussi complessità e risultati di approssimazione.

3.3.2 Il problema del NO-WAIT FLOW SHOP

Il problema del NO-WAIT FLOW SHOP è definito come segue: consideriamo un insieme di unità e un insieme di m macchine M_j (j=1,2,…,m). Ogni unità deve visitare le m macchine nello stesso ordine. L'obiettivo è quello di sequenziare le unità in modo da minimizzare il tempo di completamento, col vincolo che ogni unità, una volta iniziata, può trascorrere nel sistema solo il tempo strettamente necessario alla sua lavorazione, ossia non sono permesse attese tra le macchine (Fig.3.12). I tempi di set–up tra unità diverse possono considerarsi nulli, come pure i tempi di trasporto tra un centro e l'altro.

Le unità appartengono a b differenti part-types, ognuno consistente in un insieme di unità identiche. Nel seguito, le lettere maiuscole indicheranno i part type, quelle minuscole le unità: ad esempio, k indica un'unità di tipo K. Per semplicità, spesso nel seguito diremo "un'unità k" invece che "un'unità di tipo K". Il tempo richiesto da un'unità k sulla macchina M_j verrà indicato con K_j. Sia q_k il numero di pezzi di tipo K.

NO-WAIT FLOW SHOP è un caso particolare di TRAVELING SALESMAN PROBLEM (Reddi e Ramamoorthy 1972, Wismer 1972). Infatti, consideriamo un grafo orientato completo $G=(P,A)$ in cui P corrisponde all'insieme di tutte le unità ($|P|=\Sigma_k q_k$), $|A|=|P|(|P|-1)$ e un arco (h,k) è pesato con il *costo* di processare un'unità di tipo k *immediatamente dopo* un'unità h, con il vincolo di no-wait. Tale costo è dato da:

$$C(H,K)= H_1 + \max \{ 0 ; H_2 - K_1 ; (H_2 + H_3) - (K_1 + K_2) ; ...;$$

$$(H_2 + H_3 + \ldots + H_m) - (K_1 + K_2 + ... + K_{m-1})\}$$

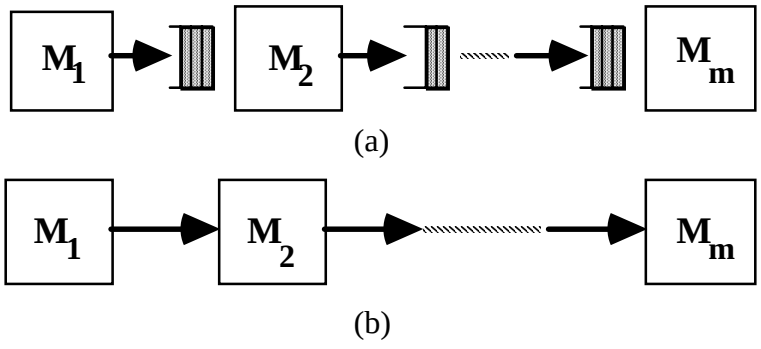
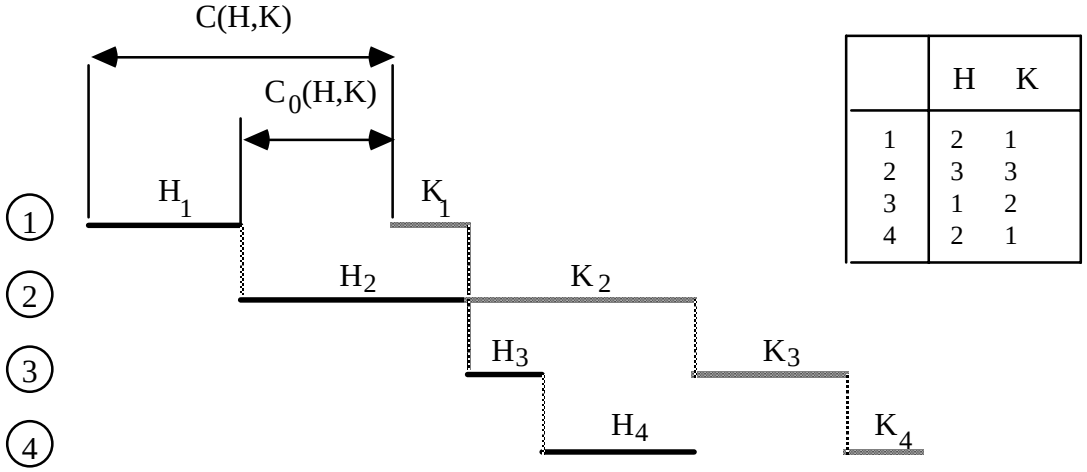


Figura 3.12. (a) Flow Shop (b) No-Wait Flow Shop.

secondo questa definizione, se un'unità k è processata in cascata a una di tipo h, $C(H,K)$ è pari alla lunghezza dell'intervallo che separa gli istanti d'inizio delle due unità (Fig.3.13). La funzione $C(\supseteq,\supseteq)$ è asimmetrica ma rispetta la disuguaglianza triangolare; questa caratteristica è sfruttata nei risultati presentati in seguito. Un'altra utile funzione è:

$$C_0(H,K) = C(H,K)-H_1$$

questa quantità rappresenta il tempo che la prima macchina trascorre in attesa tra il completamento di un'unità h e l'inizio di un'unità k (Fig.3.13).



	H	K
1	2	1
2	3	3
3	1	2
4	2	1

Figura 3.13. Definizione di $C(H,K)$ e $C_0(H,K)$

Una soluzione ammissibile al problema di TSP specifica un ciclo, cioè una sequenza "chiusa" di unità, nel senso che al costo complessivo contribuisce anche il tempo che intercorre tra il completamento dell'ultima unità e l'inizio della prima. Per determinare una sequenza "aperta", ovvero che specifica anche quale dev'essere la prima e quale l'ultima unità, basta considerare un'unità fittizia *y*, caratterizzata dall'avere tempo di processamento nullo su tutte le macchine (Wismer 1972). In tal caso, il problema può ancora formularsi come ciclo hamiltoniano di costo minimo: l'unità precedente (seguente) *y* nella soluzione ottima sarà l'ultima (la prima) unità della sequenza.

In definitiva, il problema di NO-WAIT FLOW SHOP può esprimersi come segue:

Problema P1

Dati

un insieme $M=\{M_1,M_2,\ldots,M_m\}$ di macchine, un insieme $\Lambda=\{A,B,\ldots\}$ di s part type di dimensioni $q_a,q_b,\ldots$ e una matrice di tempi $T=\{X_j\}$, ove X_j è il tempo richiesto da unu'unità x sulla macchina M_j

trovare

il ciclo hamiltoniano su un grafo orientato completo G=(P,A) in cui l'insieme dei nodi P corrisponde all'insieme di tutte le unità e il costo di un arco da un'unità di tipo h ad una di tipo k è pari a C(H,K).

tale che

il costo del ciclo sia minimo

---

f (f*) indicherà il valore di una soluzione ammissibile (ottima), e S (S*) la sequenza che consente di ottenere f (f*) .

Il problema *P1* può risolversi in tempo polinomiale per m=2 (Gilmore e Gomory 1964), mentre per tre o più macchine il problema è NP–completo (Papadimitriou e Kanellakis 1980, Röck 1984). Diversi casi particolari sono stati analizzati da Panwalkar e Woollam (1979,1980) e Röck (1982): in tali casi, i tempi di processamento soddisfano particolari condizioni e questo consente di risolvere i problemi in modo efficiente. Kanellakis e Papadimitriou (1980) propongono una euristica di ricerca locale per il TSP asimmetrico prendendo come problema–test il NO-WAIT FLOW SHOP (con m=4). Per il caso generale con m macchine, Röck e

Schmidt (1983) hanno proposto un'euristica che nel caso peggiore dà luogo a un errore relativo pari a ($\cup m/2' - 1$).

In questo capitolo vedremo come il fatto che molte unità sono presenti in copia multipla possa essere sfruttato al fine di ottenere algoritmi efficienti, con un'approssimazione abbastanza piccola, che migliora al crescere della cardinalità dei part type. Dopo alcuni risultati preliminari (§3.3.3), vedremo (§3.3.4) un algoritmo approssimato, in cui l'errore relativo tende a zero al crescere della dimensione di qualunque part type; nel cso di unità tutte diverse (q_k=1 per ogni k), l'approssimazione dell'algoritmo è paragonabile a quella dell'algoritmo di Röck e Schmidt. La complessità dell'algoritmo è quella di un algoritmo risolutivo di un problema di trasporti con b sorgenti e b destinazioni.

Una soluzione ammissibile per *P1* consiste di un ciclo (hamiltoniano) sul grafo *G*. L'idea che sta dietro l'algoritmo approssimato che vedremo è quella di "fondere" opportunamente un insieme di sottocicli, ciascuno dei quali tocca un sottoinsieme dell'insieme *P* dei nodi. Se tutti i part type hanno la stessa cardinalità, questi sottocicli sono quelli della soluzione ottima del problema di assegnamento che si ottiene rilassando i vincoli di "subtour elimination" del problema di TSP. A differenza di altri algoritmi a "fusione di sottocicli" (come ad esempio l'algoritmo esatto per il caso m=2), il ciclo hamiltoniano viene ottenuto fondendo simultaneamente tutti i sottocicli, e non attraverso una sequenza di fusioni a due a due.

3.3.3 Risultati generali sul NO-WAIT FLOW SHOP

3.3.3.1 *NO-WAIT FLOW SHOP e problema di trasporti*

Si consideri il seguente problema:

Problema P2

Data

un'istanza del problema P1

trovare

la soluzione ottima di un problema di trasporti derivato in cui: il numero di sorgenti e destinazioni è pari a b; la quantità prodotta (richiesta) dalla i–esima sorgente (destinazione) è pari a q_i; il costo di trasporto dalla sorgente H alla destinazione K è dato da C(H,K).

Sia η (η^*) il valore di una soluzione ammissibile (ottima) del problema *P2*; sia inoltre x_{hk}^* il numero di unità inviate dalla sorgente *H* alla destinazione *K* nella soluzione ottima.

Si noti che l'istanza del problema di trasporti introdotto ha la seguente proprietà:

Proposizione 3.1 *– Data una sequenza S per il problema P1, è sempre possibile associarvi una soluzione ammissibile del problema P2 tale che* $\eta = f$.

S = a b a c c c c d e f e f f f (a)

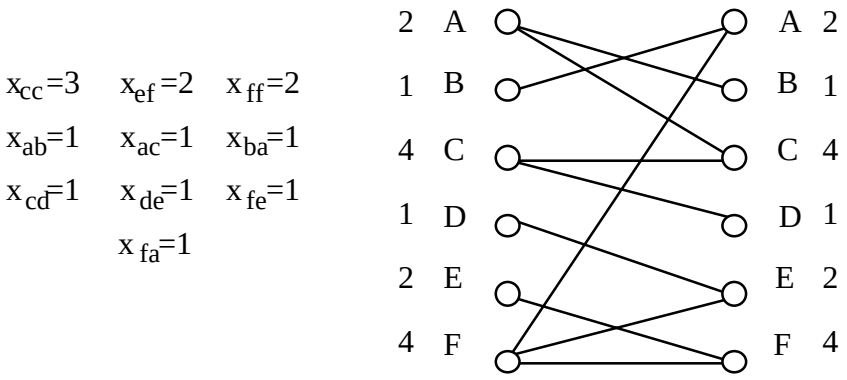


Figura 3.14. Una sequenza S e la corrispondente soluzione ammissibile di *P2*.

Infatti, data una soluzione di NO-WAIT FLOW SHOP, una soluzione ammissibile al problema di trasporti è ottenuta ponendo x_{ij} pari al numero di volte che un'unità *j* segue un'unità *i* in *S* (Fig.3.14) (ovviamente, l'inverso non è vero). Conseguenza immediata di questo fatto è che $\eta^* = f^*$.

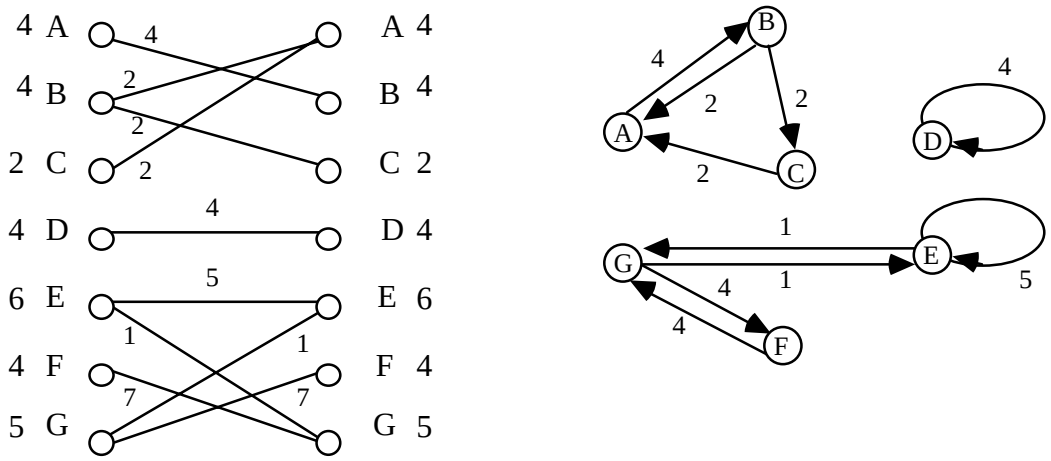
A partire da una soluzione ottima x^* di *P2*, definiamo adesso un multigrafo *G(x*)* in cui ogni nodo corrisponde a un part type e per ogni coppia (*H,K*) tale che $x_{hk}^* > 0$, vi sono x_{hk}^* archi (Fig.3.15), e ogni arco dal nodo *H* al nodo *K* è pesato con *C*(*H,K*). Il costo totale degli archi in *G(x*)* è perciò dato da

$$\sum_{(H,K)\in \Lambda\times\Lambda} C(H,K)\, x_{hk}^* = \eta^*$$

Si noti che per ogni nodo *H* in *G(x*)*, il numero di archi entranti in *H* è pari al numero di archi uscenti da *H*: quindi, *ogni componente connessa di G(x*) è euleriana.*

Supponiamo che *G(x*)* consista di una sola componente connessa. Dal momento che è euleriano, è possibile visitare tutti gli archi esattamente una volta e tornare al nodo di partenza: ciò mostra che ogni ciclo euleriano su *G(x*)* definisce un

sequenziamento *S*; il corrispondente valore f è dato dalla lunghezza totale del ciclo euleriano, ossia η^*. Dunque, per la Proposizione 3.1, tale sequenziamento è ottimo, ossia $S=S^*$. Al contrario, supponiamo che ci sia un sequenziamento ottimo S^* tale che $f^*=\eta^*$: S^* definisce naturalmente un ciclo euleriano su $G(x^*)$, che dev'essere perciò connesso. Perciò, vale il seguente teorema:



(a) (b)

k	1	2	3	4	5	6
Φ_k	{A B}	{A B C}	{D}	{E}	{E G}	{F G}
$q^{[k]}$	2	2	4	5	1	7

(c)

Figura 3.15. (a) una soluzione ottima x^* di *P2*

(b) il multigrafo $G(x^*)$ (i numeri indicano la molteplicità degli archi)

(c) l'insieme di cicli elementari.

Teorema 3.2 – *Data un'istanza di P2, esiste una soluzione ottima* x* *tale che G(x*) è connesso, se e solo se* $\eta^*=f^*$.

In altre parole, se $f^*=\eta^*$, esiste una corrispondenza biunivoca tra sequenziamenti ottimi e cicli euleriani su $G(x^*)$. E' abbastanza interessante osservare che vale la seguente condizione sufficiente perché sia $f^*=\eta^*$:

Teorema 3.3 *– Se x* non è degenere,* $\eta^*=f^*$.

DIM. - Vedi (Agnetis 1989).

Dunque, se *G(x*)* consiste di un'unica componente connessa, basta trovare un ciclo euleriano su di esso ed otteniamo un sequenziamento ottimo. Viceversa, quando tale multigrafo non è connesso, non è possibile "saltare" da una componente connessa all'altra di *G(x*)* senza pagare qualcosa in più rispetto a η^*. L'idea, allora, che si può pensare di adottare è in qualche modo quella di riuscire a *fondere* le componenti connesse di *G(x*)* in un modo che garantisca un aumento non eccessivo nella funzione obiettivo. Nel seguito presenteremo solo le idee principali; i dettagli e le dimostrazioni sono in (Agnetis 1989).

3.3.3.2 *Cicli euleriani, sottosequenziamenti e patching*

Dal momento che ciascuna componente connessa di *G(x*)* è euleriana, possiamo associare a ciascuna di esse un ciclo euleriano, che prenderà il nome di *sottosequenziamento*; sia σ_i quello associato alla *i*–esima componente connessa di *G(x*)*.

Il noto metodo per trovare un ciclo euleriano consiste nel calcolare un insieme di cicli semplici: dal momento che, per ogni ciclo, ce n'è sempre almeno un altro tale che l'intersezione degli insiemi di nodi toccati dai due cicli è non nulla, i due cicli possono essere fusi, originandone uno più grande, e questo può ripetersi fino a ottenere un unico ciclo euleriano. Un ciclo euleriano può allora sempre pensarsi come ottenuto per fusione di un certo numero di cicli *semplici* (ossia, in cui non si passa due volte per lo stesso nodo); chiamiamo tali cicli *elementari*.

Una volta scelto un σ_i per ogni $i=1,2,\ldots,n$, sia $\sigma=\{\sigma_1,\sigma_2,\ldots,\sigma_n\}$. Siccome σ_i può ottenersi per fusione di un certo insieme di cicli elementari, diremo che σ_i *contiene* quei cicli elementari. Siccome peraltro *G(x*)* è un multigrafo, uno stesso ciclo elementare può comparire più volte in un ciclo euleriano, per cui ogni ciclo elementare Φ_i ha associata una sua *molteplicità* $q^{[i]}$ (Fig.3.15(c)); sia invece il *periodo* il costo totale di un ciclo elementare (contato una sola volta):

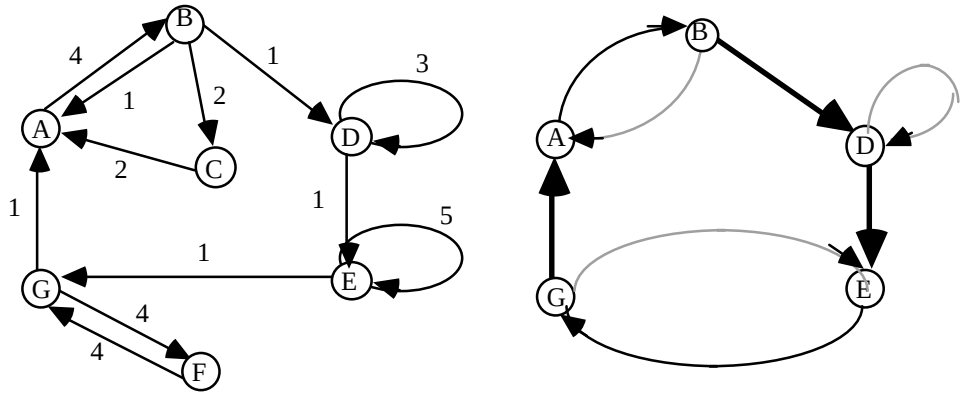
$$\tau_i = \sum_{(H,K)\in E(\Phi_i)} C(H,K)$$

Una volta calcolato un insieme Φ di cicli elementari, associamo alla *k*–esima componenete connessa di *G(x*)* il valore del più lungo periodo tra quelli dei cicli elementari contenuti in essa, sia T_k il valore di tale periodo.

Si noti che, in generale, lo stesso part type può apparire in diversi cicli elementari — ovviamente, appartenenti alla stessa componente connessa di *G(x*)*.

Il numero di cicli elementari è $O(b)$, ognuno con $O(b)$ nodi. Ricordando che in *G(x*)* ci sono esattamente q archi, il tempo necessario a calcolare un ciclo euleriano su *G(x*)* è $O(\min(b^2, q))$.

A questo punto, possiamo definire l'operazione che consiste nel connettere le diverse componenti connesse di *G(x*)*, attraverso l'inserimento di n nuovi archi e la rimozione di altrettanti archi; questo renderà il nuovo multigrafo euleriano e potremo dunque associare un sequenziamento di costo pari alla somma dei pesi dei suoi archi. Precisamente, indichiamo con *patching P(E'∅E")* l'operazione consistente nel sostituire un insieme *E'* di n archi (uno da ogni componente connessa) con un insieme *E"*, tale che il multigrafo risultante *G'(x*)* sia euleriano (e quindi connesso). Siano *c(E')* e *c(E")* il costo totale degli archi in *E'* e *E"* rispettivamente (Fig.3.16). La quantità *F(E'∅E") =c(E")–c(E')* rappresenta il *costo di patching.*



(a) (b)

S = a b c a b c a b a b d d d d e e e e e e g f g f g f g f g (a)

(c)

Figura 3.16. Patching *P(E'∅E")* del grafo *G(x*)* in Fig.3.15:

E'={(B,A) (D,D) (G,E)}; E"={(B,D) (D,E) (G,A)}

(a) grafo *G'(x*)* ottenuto da *G(x*)* dopo il patching - (b) il patching tra i cicli elementari (Φ_1, Φ_3 e Φ_5)- (c) un sequenziamento definito da *G'(x*)*.

Il costo di patching *F(E'∅E")* può esprimersi in funzione dei valori $C_0(\supseteq,\supseteq)$. Ad esempio, nel caso del patching in Fig.3.16 abbiamo:

$$c(E'') - c(E') = [C(B,D)+C(D,E)+C(G,A)] - [C(B,A)+C(D,D)+C(G,E)] =$$

$$[C_0(B,D)+C_0(D,E)+C_0(G,A)+B_1+D_1+G_1] -$$
$$[C_0(B,A)+C_0(D,D)+C_0(G,E)+B_1+D_1+G_1] =$$

$$[C_0(B,D)+C_0(D,E)+C_0(G,A)] - [C_0(B,A)+C_0(D,D)+C_0(G,E)].$$

Supponiamo che $G(x^*)$ non sia connesso, e consideriamo un patching $P(E'\varnothing E'')$: siccome il multigrafo $G'(x^*)$ risultante è euleriano, è possibile associarvi un sequenziamento S, il cui valore è dato da:

$$f = \eta^* + F(E'\varnothing E'')$$

L'algoritmo approssimato presentato nel §3.3.4 consiste nel costruire un patching $P(E'\varnothing E'')$ delle componenti di $G(x^*)$ di costo $F(E'\varnothing E'')$ non troppo elevato.

3.3.4 Un algoritmo approssimato per il NO-WAIT FLOW SHOP

Nel seguito, presenteremo un algoritmo asintoticamente esatto per il NO-WAIT FLOW SHOP, nel senso che l'errore va a zero al crescere della cardinalità di qualunque part type.

Introduciamo altri due problemi collegati al problema *P2*:

Problema P3

Data

un'istanza del problema P2

trovare

la soluzione ottima di un problema di assegnamento che si ottiene da P2 supponendo che $q_i=1$ per ogni $i=1,\ldots,b$.

Sia A* il valore della soluzione ottima di P3, e z_{ij}* il valore all'ottimo delle sue variabili.

Problema P4

Data

un'istanza del problema P2

---

trovare

la soluzione ottima di un problema di trasporti che si ottiene da P2 sostituendo q_i con $(q_i–1)$, per ogni i=1,2,…,b.

Sia η_{-1}* il valore della soluzione ottima di P4, e $\underline{x}_{ij}$* il valore all'ottimo delle sue variabili.

Se q_i=1, la *i*–esima coppia sorgente/destinazione di *P2* non compare in *P4*. Si può vedere facilmente che la soluzione ottima di *P2* non può eccedere la somma delle soluzioni ottime di *P3* e *P4*, ossia vale il seguente lemma:

Lemma 3.1 *– $\eta^* = \eta_{-1}^* + A^*$.*

DIM. - Se, per ogni (*H*,*K*), poniamo $x_{hk} = \underline{x}_{hk}^* + z_{hk}^*$, otteniamo una soluzione ammissibile (in generale non ottima) al problema *P2*, il cui costo uguaglia la somma delle soluzioni ottime di *P3* e *P4*.¶

Ricordiamo che, dato *G(x*)*, che consiste di *n* componenti connesse, T_k è il più lungo dei periodi elementari contenuti nella *k*–esima componente connessa. Supponiamo allora di ordinare i sottosequenziamenti σ_i secondo la seguente

Regola di ordinamento 3.1*– Un sottosequenziamento σ_i precede un altro σ_j se $T_i = T_j$. In caso di parità si può scegliere a piacere.*

Possiamo allora scrivere, secondo tale ordinamento:

$$T_1 = T_2 = \ldots = T_n$$

d'ora in poi, σ_1 indicherà il sottosequenziamento in $\boldsymbol{\sigma}$ contenente il Φ_k più lungo; σ_2 quello in ($\boldsymbol{\sigma}$–σ_1) contenente il Φ_k più lungo; …; σ_i il sottosequenziamento in ($\boldsymbol{\sigma}$–σ_1–…–σ_{i-1}) contenente il Φ_k più lungo; e così via. Il minimo valore di $C_0(X,Y)$ quando *X* è un part type appartenente a σ_1 mentre *Y* non appartiene a σ_1 verrà chiamato *minimo costo di switch da* σ_1 e sarà indicato con *C*, ossia

$$C = \min \{ C_0(X,Y) \mid X \in \sigma_1 , Y \notin \sigma_1 \}$$

Può essere a questo punto mostrato che vale il seguente lower bound per *f**, per la cui dimostrazione si rimanda a (Agnetis 1989):

Lemma 3.2 *– $f^* = C + \eta_{-1}^*$.*

Vediamo ora come costruire un patching di costo limitato. Verrà di seguito riportata solo la descrizione dell'algoritmo; per la dimostrazione della sua correttezza sarebbero necessari altri teoremi e lemmi che sono contenuti in (Agnetis 1989). Ricordiamo che, come al solito, *n* indicherà il numero di componenti connesse di *G(x*)*, e dunque di sottosequenziamenti.

Supponiamo di avere risolto *P2*, generato il multigrafo *G(x*)*, calcolati i cicli elementari e un sottosequenziamento per ogni componente connessa di *G(x*)*, ordinato i σ_i secondo la regola 3.1. Nel seguito, indichiamo con *U* e *V* due part type, $U \in \sigma_1$ e $V \in \sigma_1$ per i quali si ottiene il minimo costo di switch da σ_1, cioè tali che $C = C_0(U,V)$, e siano Φ_1 e Φ_χ due cicli elementari cui *U* e *V* appartengono rispettivamente (sia σ_χ il sottosequenziamento contenente Φ_χ). Sia *Z* il part type che segue *U* in Φ_1, mentre *W* il part type che precede *V* in Φ_χ . Per ogni $\sigma_i ? \sigma_1$ e $\sigma_i ? \sigma_\chi$, scegliamo arbitrariamente un ciclo elementare (uno da ogni sottosequenziamento): sia Φ_i il ciclo elementare scelto dal sottosequenziamento σ_i. Dal §3.3.3.2, ricordiamo che $\tau_i = T_i$ per ogni *i*. I cicli $\{\Phi_1, \Phi_2, \ldots, \Phi_n\}$ sono i cicli elementari coinvolti nel patching che stiamo per specificare. Nelle figure riportiamo solo i cicli elementari, ma è sottinteso che che quando si passa da un ciclo elementare all'altro, l'intero sottosequenziamento a cui quel ciclo elementare appartiene viene eseguito prima di passare su un nuovo ciclo elementare.

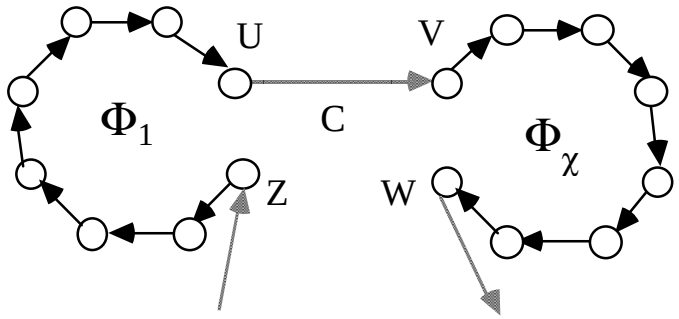


Figura 3.17. La prima parte del patching.

Supponiamo di iniziare il sequenziamento da un'unità di tipo *z*; in particolare, tra le unità *z* in σ_1, ne scegliamo una preceduta da un'unità di tipo *u*. Dopodiché, si prosegue secondo il sottosequenziamento σ_1: l'ultima unità processata in σ_1 è di tipo *u*: quindi, il primo arco dell'insieme *E'* è un arco (*U,Z*). Dobbiamo ora decidere a quale sottosequenziamento passare: scegliamo σ_χ, iniziando da un'unità di tipo *v* (ossia, inseriamo l'arco (*U,V*) in *E''*); in particolare, tra le unità di tipo v in σ_χ , ne scegliamo una preceduta da un'unità *w*. Dopo l'unità *v*, si segue il sottosequenziamento σ_χ: l'ultima unità a essere processata in σ_χ sarà di tipo *w* (Fig.3.17).

Fin qui, avendo rimosso l'arco (*U,Z*) e aggiunto (*U,V*), l'aumento nel valore di *f* rispetto a η^* è dato da *C(U,V)=C*. A questo punto la costruzione del patching prosegue in modo leggermente diverso asseconda che *n* sia pari o dispari. Ad ogni modo, il sottosequenziamento da eseguire sarà $\sigma_{\chi+1}$ o $\sigma_{\chi-1}$ asseconda che χ sia a sua volta pari o dispari; dopodiché, si eseguono gli altri sottosequenziamenti (senza più badare alle unità che si interfacciano) secondo l'ordinamento 2.1, ovviamente saltando i sottosequenziamenti già processati. La struttura del patching che si ottiene nei vari casi è illustrata nelle Figg. 3.18, 3.19 e 3.20. In tali figure, per alcuni gruppi di archi del patching è indicato un upper bound sul loro costo; grazie a questi upper bound, la cui dimostrazione, piuttosto laboriosa, è riportata in (Agnetis 1989), è possibile dimostrare i risultati di approssimazione riportati nel seguito.

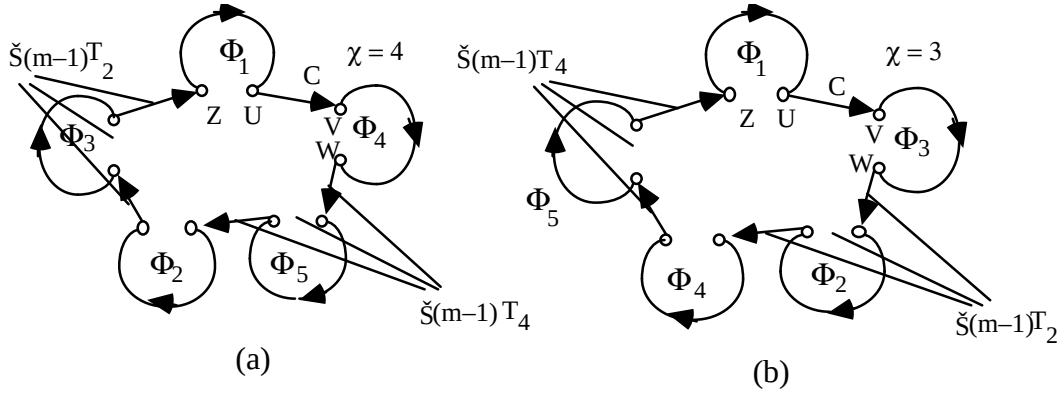


Figura 3.18. Patching tra cicli elementari degli *n* sottosequenziamenti quando *n* è dispari:

(a) χ è pari, (b) χ è dispari

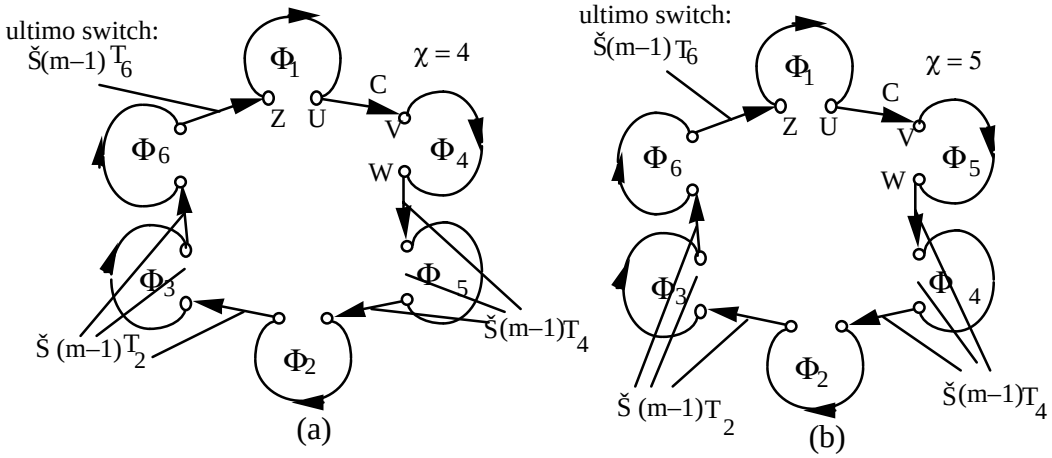


Figura 3.19. Patching quando *n* è pari e χ?*n*: (a) χ è pari, (b) χ è dispari.

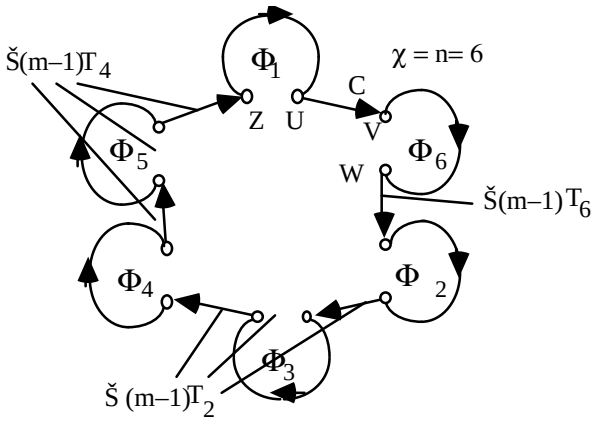


Figura 3.20. Patching quando $\chi=n$.

Scelto un insieme di n cicli elementari, uno da ogni componente connessa $\Phi_1,\Phi_2,...,\Phi_n$ (Φ_i indica il ciclo contenuto in σ_i) e due part type U e V tali che $U \in \Phi_1$, $V \in \sigma_i$ e $C_0(U,V)=C$, la seguente procedura PATCH calcola gli insiemi di archi E' e E'' che definiscono un patching $P(E' \varnothing E'')$:

procedure PATCH (***input:*** $\Phi_1,\Phi_{2,...,}\Phi_n;U,V$; ***output:*** E',E'');

begin

if (n è dispari) **then** consideriamo le seguenti coppie di cicli elementari:

$$P := \{[\Phi_2,\Phi_3]; [\Phi_4,\Phi_5]; \ldots; [\Phi_{n-3},\Phi_{n-2}];[\Phi_{n-1},\Phi_n]\}$$

if (n è pari) **then** consideriamo le seguenti coppie di cicli elementari:

$$P := \{[\Phi_2,\Phi_3]; [\Phi_4,\Phi_5]; \ldots; [\Phi_{n-2},\Phi_{n-1}]\}$$

Φ_χ:= ciclo cui appartiene V;

Z:= part type tale che $(U,Z) \in E(\Phi_1)$;

W:= part type tale che $(W,V) \in E(\Phi_\chi)$;

if n=2 **then**

begin

$E' := \{(U,Z)\} \approx \{(W,V)\}$;

$E'' := \{(U,V)\} \approx \{(W,Z)\}$;

end;

if n=3 **then**

begin

if (*n* è dispari) **or** (*n* è pari e χ?*n*) **then**

begin

Φ_ψ:= ciclo accoppiato a Φ_χ in P;

consideriamo il seguente ordinamenti dei cicli elementari:

R := [$\Phi_1,\Phi_\chi,\Phi_\psi,\Phi_2,\Phi_3,...,\Phi_{n-1},\Phi_n$];

end

if (*n* è pari e χ=*n*) **then**

begin

consideriamo il seguente ordinamenti dei cicli elementari:

R := [$\Phi_1,\Phi_n,\Phi_2,\Phi_3,...,\Phi_{n-1}$];

end;

sia γ_i l'*i*–esimo ciclo in *R*;

H^1:=*U*; K^1:=*Z*;

H^2:=*W*; K^2:=*V*;

for *i*:=3 a *n* **do** sia (H^i,K^i) un arco appartente a $E(\gamma_i)$;

E':={ $\approx_{i=1,...,n}$ (H^i,K^i)};

E'':={ $\approx_{i=1,...,n-1}$ (H^i,K^{i+1})]} ≈ {(H^n,K^1)}

end

end PATCH.

Ricapitolando, può quindi proporsi il seguente algoritmo:

algorithm SCHEDULING (***input***: un'istanza di *P1*; ***output***: un sequenziamento *S*);

begin

Risolvere il problema di trasporti (P2) associato al TSP (P1); sia $\{x^*\}$ la soluzione ottima;

generare il multigrafo $G(x^*)$; sia n il numero di componenti connesse in $G(x^*)$;

for i:=1 to n **do** calcolare un ciclo euleriano (sottosequenziamento) sulla i-esima componente connessa di $G(x^*)$;

ordinare i sottosequenziamenti secondo la regola 3.1; sia σ_h l'h–simo sottosequenziamento;

if (n=1) **then** processare nell'ordine specificato dal sottosequenziamento

else begin

selezionare i part type U e V tali che $C_0(U,V) := \min \{C_0(H,K) \mid H \ \sigma_1$ e $K \ \sigma_1\}$;

σ_χ := sottosequenziamento contenente il part type V;

Φ_1:= ciclo elementare contenuto in σ_1 e contenente il part type U;

Φ_χ:= ciclo elementare contenuto in σ_χ e contenente il part type V;

for i:=2 **to** n **do if** $i?\chi$ **then** selezionare un ciclo Φ_i contenuto in σ_i ;

PATCH [$\Phi_1,\Phi_2,\Phi_3,\ldots,\Phi_n$; U,V; E',E''];

$G'(x^*) := G(x^*) \approx E'' - E'$;

S := Ciclo euleriano su $G'(x^*)$

end

end SCHEDULING.

3.3.5 Approssimazione e complessità dell'algoritmo

Si può dimostrare (Agnetis 1989) che il costo f associato al grafo euleriano che si ottiene dopo avere effettuato un patching secondo la procedura illustrata è limitato superiormente come segue:

$$f^* = \eta^* + C + \frac{(m-1)A^*}{2}$$

A partire da questo limite superiore, è possibile allora ottenere un limite superiore all'errore relativo che si commette utilizzando l'algoritmo SCHEDULING. Vanno distinti due casi, asseconda che sia $C = A^*$ oppure $C < A^*$: nel primo caso si utilizza come lower bound per f^* quello fornito dal Lemma 3.2, nel secondo quello fornito dalla Proposizione 3.1. In definitiva si ottiene:

Teorema 3.4 *-L'errore relativo che si commette usando l'algoritmo SCHEDULING non è superiore a* $\frac{m+1}{2}\frac{A^*}{\eta^*}$.

Siccome si può vedere facilmente che η^* è strettamente crescente rispetto a qualunque q_i, possiamo concludere che l'algoritmo SCHEDULING è asintoticamente esatto, nel senso che l'errore va a zero al crescere della cardinalità di qualsiasi part type. Si noti che se tutti i part type hanno la stessa cardinalità k ($q=kb$), allora è $\eta^*=kA^*$ e quindi

$$\varepsilon = \textbf{Errore.}$$

in questo caso ogni componente connessa di $G(x^*)$ contiene esattamente un ciclo elementare, e tutti i cicli elementari hanno la stessa molteplicità k. Questa espressione può trovare utile impiego in quei casi in cui k non è dato, ma è una variabile decisionale: una volta fissato un valore ε^*, può calcolarsi il minimo valore di k per cui tale l'errore massimo è senz'altro inferiore a ε^*.

Per quanto riguarda la complessità dell'algoritmo SCHEDULING, è facile riconoscere che essa è dominata dal primo passo, ossia dalla soluzione del problema *P2*. Infatti, il calcolo dei sottosequenziamenti, il calcolo del minimo costo di switch da σ_1, l'ordinamento dei sottosequenziamenti e il patching stesso possono essere effettuati in $O(b^2)$, $O(b^2)$, $O(b \log b)$ e $O(b)$ rispettivamente. Attualmente, algoritmi *fortemente polinomiali* sono disponibili per il problema dei trasporti (ad esempio, quello di Tardos (1985)): questo vuol dire che possiamo esprimere la complessità del nostro approccio con un polinomio nel solo *numero* di part type, e non nella loro cardinalità. Comunque, da un punto di vista pratico, va ricordato che il metodo del simplesso applicato al problema dei trasporti è in genere più efficiente, benché possa avere complessità esponenziale nel caso peggiore.

	M_1	M_2	M_3	M_4	q_i
A	5	2	5	6	1
B	2	6	6	2	1
C	7	5	3	2	3
D	5	3	3	1	2
E	1	1	1	1	1
F	3	1	1	5	2
G	3	2	2	1	2

(a)

	A	B	C	D	E	F	G
A	1	0	0	2	10	8	6
B	5	4	0	4	11	9	7
C	1	3	0	0	7	5	3
D	0	1	0	0	4	2	1
E	0	0	0	0	0	0	0
F	0	0	0	0	4	2	0
G	0	0	0	0	2	0	0

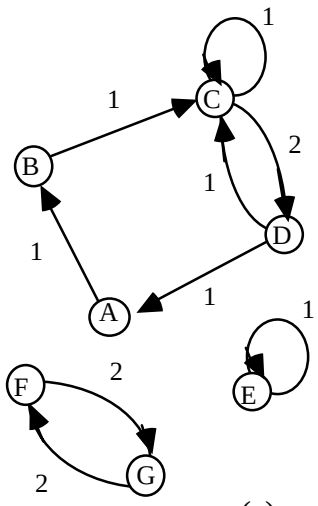
(b)

Figura 3.21. Esempio: (a) tempi di processamento (b) i costi $C_0(\trianglerighteq,\trianglerighteq)$.

3.3.6 Esempio

Applichiamo l'algoritmo SCHEDULING al problema in Fig.3.21. In questo caso unità di b=7 tipi diversi devono essere sequenziate su m=4 macchine. La soluzione ottima $\{x^*\}$ del problema *P2* dà luogo al multigrafo $G(x^*)$ con n=3 componenti connesse, come illustrato nella Fig.3.22(a); i cinque cicli elementari sono indicati in Fig.3.22(b). Siccome Φ_1 è il ciclo elementare più lungo, σ_1 è il sottosequenziamento associato alla componente che contiene Φ_1, Φ_2 e Φ_3; σ_2 conterrà Φ_5 mentre σ_3 conterrà Φ_4.

Dalle Figg.3.22 e 3.23 osserviamo che C=1 e si ha in corrispondenza dell'arco (D,G); quindi χ=2. Scegliamo Φ_1 σ_1, Φ_5 σ_2, and Φ_4 σ_3 come cicli elementari che prendono parte al patching: si ottiene così il multigrafo $G'(x^*)$ in Fig.3.23(a); se si sceglieva Φ_2 invece di Φ_1 (cosa altrettanto lecita in quanto ambedue tali cicli elementari contengono il part type D), si sarebbe ottenuto il multigrafo $G''(x^*)$ in Fig.3.23(b). I sequenziamenti corrispondenti a queste due possibili soluzioni sono S' e S'' , pure mostrati in Fig.3.23. Si noti che in ambedue i casi la durata del sequenziamento è f=56. Si noti che queste *non* sono le migliori soluzioni cui un patching potesse dare luogo: scegliendo il patching $P(E'\varnothing E'')$ con E'={$(D,A),(G,F),(E,E)$} e E''= {(D,F), (G,E), (E,A)} si otterrebbe f=55. In questo esempio, non era possibile trovare il sequenziamento ottimo attraverso un patching: infatti, esso è dato da:



(a)

$\eta^*=51$

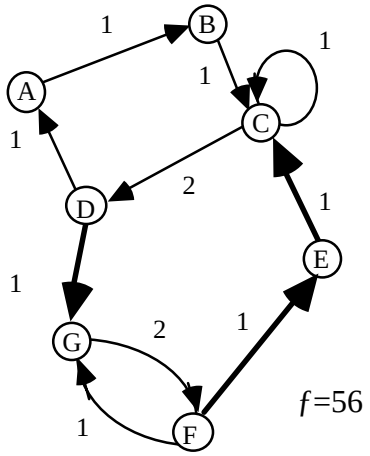
i	1	2	3	4	5
Φ_i	ABCD	CD	C	E	FG
τ_i	19	12	7	1	6
$q^{[i]}$	1	1	1	1	2

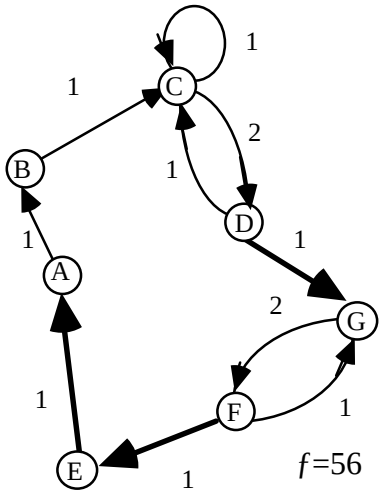
(b)

Figura 3.22. (a) Il multigrafo *G(x*)* - (b) i cicli elementari.

$$S^* = a\ b\ c\ c\ d\ c\ d\ g\ f\ g\ e\ f\ (a)$$

e f^*=54. Si noti che l'errore relativo di *S'* è pari a $f - f^*/f = 0.037$, molto più piccolo del limite teorico del Teorema 3.4: $(m+1)A^* / 2\eta^* = 1.27$.





E' = {(D,C);(F,G);(E,E)}
E'' = {(D,G);(F,E);(E,C)}

S' = a b c c d g f g f e c d (a)

(a)

E' = {(D,A);(F,G);(E,E)}
E'' = {(D,G);(F,E);(E,A)}

S''= a b c c d c d g f g f e (a)

(b)

Figura 3.23. Multigrafi *G'(x*)* e *G''(x*)* ottenuti applicando

l'algoritmo SCHEDULING con due diverse scelte del patching *P(E'∅E'')*

3.4 Selezione dei part type e sequenziamento in un pipeline di due macchine con buffer limitati

3.4.1 Introduzione

In questo capitolo presentiamo un nuovo approccio ad un problema decisionale più ampio di quello analizzato nel §3.3: precisamente, trattiamo del problema integrato della *selezione dei part type* e del *sequenziamento di lotti e pezzi* in un sistema flessibile pipeline costituito da due macchine. Si è cioè formulato un modello per orientare le scelte relative alla selezione dei part type in base all'uso congiunto che, della risorsa di immagazzinamento, verrebbe fatta da insiemi di part type, qualora vengano messi in produzione contemporaneamente.

L'approccio consiste nell'avere, in ogni istante, in lavorazione un numero ristretto di part type, in opportuni *rapporti relativi* di mix, in modo tale da bilanciare il carico di lavoro tra le macchine, tenendo conto della limitata capacità dei buffer tra le due macchine. Perché ciò accada è però anche necessario avere una regola di sequenziamento dei pezzi che consenta, all'interno del periodo di lavorazione definito da un certo mix, di poter conseguire il massimo della utilizzazione delle due macchine, senza eccedere mai la capacità del magazzino di semilavorati. Come si vedrà nel §3.4.3, per il sistema in esame tale regola effettivamente esiste ed è molto semplice.

A ciascun intervallo di tempo durante il quale il mix produttivo rimane costante corrisponde un *lotto*. Noi supporremo che la produzione sia organizzata in lotti di *due* part type ciascuno. Questo perché, come vedremo, due part type sono sufficienti (se non esistono vincoli molto stringenti sul loro rapporto relativo) a bilanciare i carichi di lavoro tra le macchine, e si può esprimere in modo quantitativo la capacità minima del magazzino necessaria per mantenere le macchine al massimo della loro utilizzazione, allorché una certa *coppia* di part type è in produzione.

Dunque, la produzione è organizzata in lotti di due part type ciascuno. All'interno di ciascun lotto, i due part type sono lavorati in proporzioni legate all'utilizzo delle risorse del sistema, e che non dipendono cioè dalla quantità che dei vari part type è richiesta dal mercato, ossia dall'effettiva domanda dei vari tipi di pezzi. E' allora necessario formulare un modello che consenta di suddividere la produzione richiesta in lotti, ove in generale lo stesso part type sarà presente in due o più lotti. Il problema di *part type selection* (*PTS*) consiste proprio nello specificare la composizione di questi lotti. Peraltro, siccome ogni operazione che ciascun part type deve subire richiede che certi utensili siano caricati nei corrispondenti magazzini utensili delle due macchine, alcune coppie di part type saranno a priori *incompatibili*, se cioè l'unione degli insiemi di utensili richiesti dai due part type eccede la capacità dei magazzini utensili.

Una volta determinata la composizione dei vari lotti, occorre decidere l'ordine migliore in cui processare i lotti stessi, ossia il *sequenziamento dei lotti* (*BS*). Infatti, ogni volta che si passa da un lotto ad un altro, alcuni insiemi di utensili vanno tolti e sostituiti con altri, per cui appare ragionevole cercare quel sequenziamento che minimizza il numero totale di operazioni di cambio–utensile che è necessario effettuare.

Con il termine *input sequencing* (*IS*) indichiamo invece il problema di decidere il miglior sequenziamento con cui introdurre *i pezzi* di un lotto, che, per quanto detto prima, sono di due soli tipi diversi. Due obiettivi principali di *IS* sono: massimizzare l'utilizzo del sistema (ovvero, minimizzare i tempi di attesa delle macchine) e minimizzare il massimo livello raggiunto dal buffer che ospita i pezzi in attesa di M_2. Come abbiamo già osservato nel §3.3.1, questi due obiettivi sono contrastanti, in quanto tipicamente, per fare in modo che la seconda macchina abbia sempre qualcosa da fare (massimizzandone così l'utilizzo), si può decidere di sequenziare i pezzi in modo da "affollare" quanto più possibile il buffer tra le due macchine. Questo, peraltro, è proprio ciò che fa la ben nota regola di Johnson (1952) per la minimizzazione del tempo di completamento in un flow shop con due macchine. D'altronde, in generale trovare la sequenza di pezzi che consenta di minimizzare il tempo di completamento quando la capacità del buffer è limitata è un problema NP–completo (Papadimitriou e Kanellakis 1980). Nel §3.4.3 vedremo invece che, se il lotto consiste di due soli part type, è possibile sequenziare i pezzi in modo da avere una sequenza *ottima* dal punto di vista del tempo di completamento e tale che il massimo livello raggiunto dal buffer di semilavorati è *minimizzato*. Ecco allora che la conoscenza di tale valore permette di capire se la coppia di part type in questione è compatibile o meno con l'attuale capacità del magazzino tra le due macchine.

La strategia di gestione della produzione di breve termine presentata in questo §3.4 è basata sull'approccio flessibile (§3.4.2): i lotti sono costituiti da coppie di part type, in proporzione relativa tale da bilanciare i carichi delle due macchine, e sequenziati in modo da minimizzare la massima lunghezza raggiunta dalla coda tra le due macchine. Va detto che, avendo in ogni istante solo due part type in produzione, in pratica accade che tutte o quasi le possibili coppie di part type sono ammissibili dal punto di vista dell'occupazione dei magazzini utensili.

Il nostro approccio ha il vantaggio di essere implementabile in modo estremamente semplice, e garantisce la massima utilizzazione teorica dei centri di lavorazione. I limiti di questo approccio stanno nel fatto che esso — dal momento che fa uso di un modello di PL — è appropriato per produzione su scala medio–grande. Più precisamente, i tempi operazione e/o la cardinalità dei part type debbono essere abbastanza elevati da far sì che i riattrezzaggi non avvengano troppo di frequente.

Il nostro approccio è particolarmente utile in un contesto dinamico, caratterizzato cioè da ordini di produzione che arrivano in istanti diversi. Periodicamente, si può avere bisogno di risolvere varie volte *PTS*; per tenere conto dei nuovi ordini arrivati, che insieme a quelli già presenti, posono concorrere ad utilizzare meglio le risorse del sistema.

Dopo alcuni brevi richiami sui problemi di selezione di part type (§3.4.2), vedremo più in dettaglio l'approccio risolutivo: dapprima vedremo come può essere risolto il problema di input sequencing (§3.4.3); quindi quello di selezione dei part type (§3.4.4) e infine il problema di sequenziare i lotti (§3.4.5). Infatti, anche se dal punto di vista pratico l'ordine in cui i problemi vengono affrontati è diverso, si ha che *IS* fornisce i parametri necessari alla risoluzione di *PTS*, e l'output di quest'ultimo è l'input per *BS*. Per i dettagli dimostrativi e gli esperimenti numerici si rimanda a (Agnetis, Arbib e Stecke 1991*a*, 1991*b*).

3.4.2 Letteratura sui problemi di selezione di part type

La letteratura relativa ai problemi di selezione dei part type si può suddividere in due gruppi, relativamente ai due principali approcci che vengono impiegati: si distinguono l'approccio *a lotti* (*batching*) e quello *flessibile* (*flexible*). Nel primo caso (utilizzato tra gli altri da Whitney e Gaul (1985), Hwang (1986), Rajagopalan (1986), Sarin e Chen (1987), Stecke e Kim (1988)), tutti i part type sono partizionati in lotti disgiunti; per ogni lotto, tutti i pezzi di un particolare tipo sono prodotti, fino a esaurimento della richiesta, prima di passare ad un nuovo lotto. In questo tipo di gestione del sistema, il riattrezzaggio dei centri di lavorazione ha luogo solo alla fine dell'esecuzione di un lotto e prima dell'inizio del successivo; c'è dunque una distinzione netta tra periodi durante i quali le macchine operano sui pezzi e quelli in cui sono ferme a causa del riattrezzaggio. L'obiettivo è quello, tipicamente, di minimizzare il numero di lotti (in quanto ciò corrisponde a minimizzare il numero di riattrezzaggi), come ad esempio nel modello euristico di Hwang e Shogan (1990).

Con l'approccio *flessibile* (Stecke e Kim 1988, 1989, 1991, Stecke e Toczylowski 1989), ad ogni istante di tempo un certo insieme di part type sono in produzione, secondo prefissate proporzioni relative (*mix ratios*). Tali proporzioni relative sono tali da bilanciare i carichi di lavoro. Allorché uno dei part type viene esaurito, gli utensili che non servono più vengono tolti per far posto a quelli richiesti da un nuovo part type che viene attivato; quest'operazione non comporta, in generale, l'arresto di tutto il sistema, ma solo delle macchine interessate, rende l'attività di cambio utensili più regolarmente distribuita nel tempo, evita il sottoutilizzo del sistema che inevitabilmente si ha nei transitori con l'approccio a lotti (rispettivamente prima che il sistema raggiunga il "regime" e alla fine, quando gli ultimi pezzi del lotto tipicamente utilizzano le macchine in modo molto sbilanciato). Un nuovo part type può essere

attivato anche prima che un altro finisca, ad esempio perché è giunto un nuovo ordine di produzione, a elevata priorità. In ogni caso, quando la composizione del mix produttivo varia, vanno ricalcolati i rapporti di produzione per consentire un migliore sfruttamento delle risorse produttive. Dunque, con l'approccio flessibile, i part type non sono partizionati in lotti disgiunti, bensì in lotti sovrapposti, laddove possiamo associare un lotto ad ogni intervallo di tempo in cui un certo insieme di part type è in produzione; quando tale insieme varia, si passa a un nuovo lotto.

In generale, l'approccio flessibile dà luogo a migliori risultati in termini di produttività (Stecke e Kim 1989); tuttavia, ci sono situazioni in cui è preferibile l'approccio a lotti (Stecke 1989).

Da segnalare anche l'interessante lavoro di Ohashi e Hitomi (1990), che formulano come programmazione intera il problema di massimizzare il numero di pezzi che devono essere prodotti nel prossimo periodo; tale obiettivo è concettualmente analogo a quello proposto nel §3.4.4.

3.4.3 Il sequenziamento dei pezzi (IS)

Sia P l'insieme dei part type, e consideriamo una sua partizione in due insiemi distinti siano essi P_1 e P_2, cosicché per ogni part type $i \quad P_1$, ($k \quad P_2$) un pezzo di tipo i (di tipo k) richiede un tempo di processamento maggiore su M_1 che su M_2 (su M_2 che su M_1). E' chiaro che, dal momento che vogliamo che all'interno di ciascun lotto i carichi di lavoro siano bilanciati sulle due macchine, uno dei part type selezionati deve appartenere a P_1 e l'altro a P_2. A questo punto consideriamo allora il seguente problema di sequenziamento (*IS*):

Sequenziamento dei pezzi (IS):

Dati

due part type $i \quad P_1$, $k \quad P_2$, da produrre in quantità sufficientemente elevata (teoricamente infinita),

trovare

un sequenziamento dei pezzi

tale che

entrambe le macchine siano sempre occupate, e il massimo livello raggiunto dal buffer tra le due macchine sia minimo.

In altre parole, risolvere *IS* vuol dire disporre di una regola che consenta, dopo la esecuzione di ciascun pezzo, di decidere quale deve essere il prossimo pezzo da lavorare, pescando da due ipotetici contenitori che non si svuotano mai. Nella formulazione di *IS* si fa riferimento, infatti, a un numero infinito di pezzi dei due tipi. Ciò in realtà ovviamente non è vero, ed è necessario quindi applicare la regola di sequenziamento che risolve *IS* ad un numero finito di pezzi di due tipi diversi. Siccome l'applicazione di tale regola comporta automaticamente che i due part type

vengono processati in un certo rapporto relativo, in generale rimarrà una domanda insoddisfatta di pezzi dell'uno o dell'altro tipo. Tale domanda insoddisfatta andrà "accoppiata" con quella di un altro part type in un altro lotto, e di questo viene tenuto conto nella fase di selezione dei part type (§3.4.4).

Si consideri la seguente regola, relativa alla coppia di part type $i \quad P_1, k \quad P_2$:

Regola di sequenziamento 3.2. *Se un pezzo di tipo* $i \quad P_1$ *può essere iniziato su* M_1 *senza che ciò crei attesa su* M_2*, allora inizia un pezzo di tipo i, altrimenti inizia un pezzo di tipo k.*

In (Agnetis, Arbib e Stecke 1991*b*) è dimostrato (in modo alquanto laborioso) il seguente:

Teorema 3.5 – *Se le quantità da lavorare di tipo i e k all'interno di un lotto sono in proporzione tale da bilanciare i carichi di lavoro, la Regola 3.2 minimizza il massimo livello raggiunto dal buffer e mantiene le due macchine sempre attive.*

Nel seguito, sia $p_{ik}(p_{jk})$ la durata dell'operazione su un pezzo di tipo i (k) sulla macchina M_k, $k = 1, 2$ (supporremo che tutti i tempi siano numeri interi). La proporzione relativa di pezzi i e k tale da bilanciare i carichi di lavoro delle due macchine può essere facilmente calcolata a-priori come segue. Sia ρ_{ik} il *numero di pezzi* di tipo k che devono essere processati per ogni pezzo di tipo i al fine di bilanciare i carichi di lavoro di M_1 e M_2. In generale, ρ_{ik} non è intero, può assumere qualunque valore razionale positivo, e dipende solo dai tempi di processamento dei due tipi di pezzi. Si può facilmente mostrare che (Agnetis, Arbib e Stecke 1991*a*):

$$\rho_{ik} = (p_{k2} - p_{k1}) / (p_{i1} - p_{i1})$$

Usando la Regola 3.2 per sequenziare pezzi di tipo i e k, l'andamento nel tempo del livello del buffer tra le due macchine è periodico, e raggiunge un massimo, indicato con q_{ik}. Anche q_{ik} dipende solo dalle durate delle operazioni dei due part type, ma il suo calcolo, benché semplice, è meno banale. Un metodo molto efficiente, è indicato in (Agnetis, Arbib e Stecke 1991*b*). Alternativamente, q_{ik} può essere sempre ottenuto per mezzo di una semplice simulazione dell'applicazione della Regola 3.2.

Ad esempio, consideriamo due part types 1 and 2; le durate delle operazioni sono p_{11}=7, p_{12}=4, p_{21}=5, p_{22}=12. Dunque, il part type 1 (part type 2) appartiene a $P_1(P_2)$. Il rapporto ρ_{12} è dato da ($p_{22}–p_{21}$ / $p_{11}–p_{12}$) = 7/3 ♠ 2.33. In Fig.3.24 è mostrato il sequenziamento ottenuto applicando la Regola 3.2, in un lotto costituito da questi due part type. Supponiamo che le quantità da processare dei due tipi siano nel rapporto ρ_{12}. Il primo pezzo del sequenziamento è di tipo 2; dopodiché, due pezzi di tipo 1 possono essere immessi senza che ciò crei attesa sulla seconda macchina. A questo punto, il tempo residuo su M_2 è pari a 6: non c'è spazio per un altro pezzo di tipo 1, siccome p_{11}= 7 > 6. Quindi, viene sequenziato un pezzo di tipo 2, seguito da due di tipo 1. A questo punto, il tempo residuo su M_2 è 7. Apparentemente, c'è spazio per un

altro pezzo di tipo 1, ma osserviamo che se sequenziamo un pezzo di tipo 1, il tempo residuo sulla seconda macchina diverrebbe 4, e questo creerebbe attesa su M_2, qualunque sia il tipo del pezzo successivo. Perciò, bisogna ancora immettere un pezzo di tipo 2, seguito da tre pezzi 1. A questo punto (*Y* in Fig.3.24) notiamo che la situazione è esattamente quella che si aveva all'inizio (punto *X*), e infatti il carico di lavoro complessivo di entrambe le macchine a questo punto è pari a $64 = 3\ p_{21} + 7\ p_{11} = 3\ p_{22} + 7\ p_{12}$.

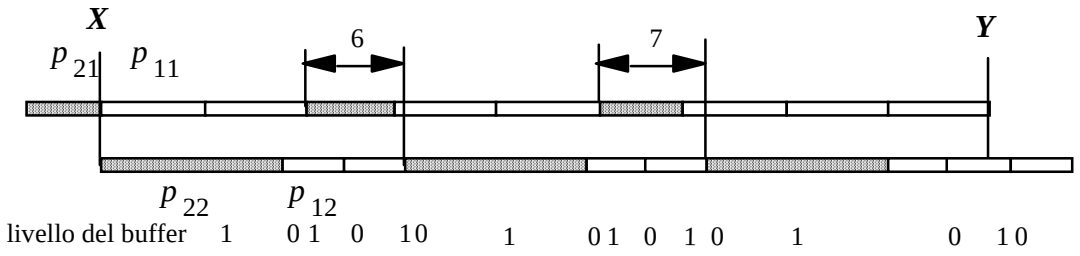


Figura 3.24. Applicazione della Regola 3.2.

Quindi, da questo punto in poi, il profilo temporale che si ottiene applicando la Regola 3.2, si ripete identicamente nel tempo. Osservando la Fig.3.24, vediamo che il livello del buffer non eccede mai 1, ossia, in quest'esempio, q_{12}=1.

Sia *q* la capacità del buffer tra le due macchine. Nel seguito, diremo che due part types *i* P_1 e *k* P_2 sono *compatibili* se l'unione degli insiemi di utensili che loro richiedono sulle due macchine non eccede la capacità dei magazzini utensili di nessuna delle due macchine, e se $q_{ik} = q$.

Si noti che rimangono fuori da P_1 e P_2 quei part type che richiedono lo stesso tempo di processamento sulle due macchine. Tali part type non pongono però particolari problemi perché i pezzi in questione possono essere sempre sequenziati consecutivamente (fino a esaurimento delle richieste), consentendo di avere un perfetto bilanciamento dei carichi di lavoro senza occupazione del buffer.

3.4.4 La selezione dei part type (PTS)

Il problema di selezione dei part type (*PTS*) consiste nello specificare la composizione di ciascun lotto, e quanti pezzi saranno lavorati in ciascun lotto, vincolando il loro rapporto relativo al valore ρ_{ik}. A causa di tale vincolo, per valori generici della domanda di ciascun tipo di pezzo, non sarà in generale garantita l'esistenza di una soluzione in cui M_2 è sempre attiva. L'obiettivo di *PTS* sarà allora quello di massimizzare la lunghezza dell'intervallo di tempo durante il quale ambedue le macchine sono occupate e la capacità del buffer non viene ecceduta.

A questo punto, l'approccio complessivo alla gestione di breve termine del sistema può meglio esprimersi come segue:

1. Elimina le coppie di part type con gruppi di utensili incompatibili.
2. Per ogni coppia rimanente (i,k), i P_1 e k P_2, calcola il rapporto ρ_{ik} e la capacità di buffer necessaria q_{ik}. Elimina le coppie per cui $q_{ik}>q$.

3. (*PTS*) Seleziona un insieme di coppie (non disgiunte) di part type (lotti), e per ogni lotto specifica il numero di pezzi da produrre; l'obiettivo è quello di massimizzare il tempo in cui la seconda macchina è attiva.
4. (*BS*) Determina l'ordine in cui i lotti devono essere processati; l'obiettivo è quello di minimizzare il numero di volte che ciascun part type viene rimesso in produzione.
5. (*IS*) Introduci i pezzi di ciascun lotto secondo la Regola 3.2.

Vediamo ora un modello per il problema indicato al passo 3 dell'approccio, mentre il passo 4 sarà brevemente discusso nel §3.4.4.

PTS può formularsi come segue:

Selezione dei part type (PTS):

Data

la domanda produttiva di ogni part type

trovare

la composizione di un insieme di lotti (non disgiunti)

tale che

sia massimo il tempo in cui ambedue le macchine sono occupate, senza eccedere la capacità del buffer.

Ricordiamo dal Teorema 3.5 che, finché i pezzi sono processati nel rapporto relativo ρ_{ik}, il buffer non eccede mai q_{ik} e ambedue le macchine sono occupate.

Introduciamo allora un grafo bipartito $G = (U, V, E)$, dove l'insieme dei nodi U (V) è in corrispondenza biunivoca con P_1 (P_2), e l'insieme E è definito come segue:

$$E + \{ (i, k) \wedge \textit{la coppia } (i, k) \textit{ è compatibile} \}.$$

Chiaramente, solo coppie di part type appartenenti a E possono essere processate nello stesso lotto. Nel seguito, useremo la seguente notazione:

— x_{ik} indica il numero di pezzi di tipo i che saranno prodotti con pezzi di tipo k nello stesso lotto. Chiaramente, il corrispondente numero di pezzi di tipo k è dato da $\rho_{ik}\, x_{ik}$.

— n_i (n_k) indica la quantità richiesta di pezzi di tipo i (di tipo k).

Una formulazione di *PTS* è allora:

$$\text{(PL1)} \qquad \max \sum_{(i;k)\ E} (p_{i1} + \rho_{ik}\, p_{k1})\, x_{ik}$$

$$\text{s.t.} \quad n_i' = \textbf{Errore.} = n_i \qquad i \quad P_1$$

$$n_k' = \sum_{i:(i;k)\ E} \rho_{ik}\, x_{ik} = n_k \qquad k \quad P_2$$

Il primo (secondo) insieme di vincoli assicura che il numero di pezzi di tipo i (k) da prodursi non eccede la richiesta per quel part type. L'obiettivo è massimizzare la

lunghezza del'intervallo durante il quale sia M_1 che M_2 sono attive, ossia, il carico parallelizzato in totale. Siccome i valori ρ_{ik} sono tali da garantire il bilanciamento dei carichi di lavoro all'interno di ciascun lotto, è sufficiente massimizzare il carico di M_1.

Ovviamente, la dimensione del lotto $(x_{ik}, \rho_{ik}x_{ik})$ ottenuta risolvendo il problema *PL1* è in genere non intera — anche se vincoliamo x_{ik} all'interezza, dal momento che in generale ρ_{ik} è non intero. Quindi, all'atto pratico sarà necessario fare uso di un'opportuna procedura di arrotondamento. Da prove sperimentali (Agnetis, Arbib e Stecke 1991*a*), risulta che le dimensioni di lotti ottenute arrotondando la soluzione ottima di *PL1* sono in genere paragonabili, in termini di qualità della soluzione, a quelle ottenute arrotondando una soluzione intera dello stesso problema. Questo è soprattutto vero nel caso di produzione di volume medio–alto, dove, in base alla nostra esperienza e per i nostri fini, gli errori di arrotondamento sono infatti trascurabili (Arbib, Lucertini e Nicolò 1991, Stecke e Kim 1989).

3.4.4.1 *I pezzi rimanenti*

In generale, come già osservato in precedenza, la soluzione ottima del problema *PL1* non consente il completo soddisfacimento di tutte le domande n_i e n_k, con $i \quad P_1$, $k \quad P_2$: solo n_i '=n_i (n_k'= n_k) pezzi di tipo $i \quad P_1$ ($k \quad P_2$) saranno inseriti in qualche lotto. Quindi, in generale occorre programmare la produzione anche dei pezzi rimanenti. Questo può non essere un problema in un contesto dinamico, in quanto gli ordini nuovi arrivati richiedono la risoluzione di un nuovo problema. Altrimenti, il problema consiste nell'assegnare le parti rimanenti ai lotti già formati, laddove ciascun assegnamento avrà un costo associato al tempo d'attesa che si va a introdurre sulla seconda macchina. Un assegnamento di pezzi rimanenti di classe P_1 (di classe P_2) può realizzarsi aggiungendo i pezzi alla fine (all'inizio) del lotto. Si può dimostrare che così facendo, il tempo di completamento del lotto è ancora ottimo (anche se, quando i pezzi residui sono di classe P_1, l'attesa della seconda macchina è inevitabile), e il massimo livello raggiunto dal buffer non eccede il valore q_{ik} che si ha quando i e k sono processati nel rapporto ρ_{ik} (Agnetis, Arbib e Stecke 1991*b*).

Un altro approccio per il problema dei pezzi rimanenti è quello di "rilassare" la definizione di coppie compatibili considerando anche alcune coppie (i, k) per cui $q_{ik} > q$. Chiaramente, il sequenziamento dei pezzi in lotti di questo tipo comportano una certa attesa sulle due macchine, dal momento che il buffer del sistema può ospitare solo q pezzi. Tuttavia, si può facilmente tenre conto in modo quantitativo di questo tempo d'attesa modificando leggermente la formulazione del problema: in particolare, possiamo imporre il vincolo che la somma dei tempi d'attesa accumulati dentro ciascun lotto non superi una certa percentuale W del tempo complessivamente

parallelizzato, ossia, del valore della funzione obiettivo. Possiamo cioè aggiungere il seguente vincolo a *PL1*:

$$\sum_{(i;k)\ E} c_{ik}\, x_{ik} \;=\; W \sum_{(i;k)\ E} (p_{i1} + \rho_{ik}\, p_{j1})\, x_{ik}$$

dove c_{ik} è il tempo totale di attesa introdotto in un lotto (i, k) unitario, ossia di cardinalità $(1, \rho_{ik})$. I valori c_{ik} possono essere calcolati agevolmente.

3.4.4.2 Una proprietà delle soluzioni di base

Vediamo ora una proprietà strutturale del problema *PL1* che riesce utile nel problema *BS* (§3.4.5). In una soluzione ammissibile di *PL1*, i valori x_{ik} specificano il numero di pezzi di tipo *i* che devono essere prodotti nello stesso lotto con $\rho_{ik}\, x_{ik}$ pezzi di tipo *k*. Data una soluzione ammissibile $\{x\}$, sia $G(x) = (U,V,E(x))$ un grafo non orientato, in cui l'insieme degli archi $E(x)$ è definito come:

$$E(x) \;+\; \{\ (i, k) \wedge i \quad P_1, k \quad P_2, x_{ik} > 0\ \}.$$

Si noti che *PL1* consiste di *n* vincoli, e dunque non più di *n* variabili possono essere non nulle in una soluzione di base.

Si può allora dimostrare il seguente teorema (Agnetis, Arbib e Stecke 1991*a*):

Teorema 3.6. Se $\{x\}$ è una soluzione di base di *PL1*, il grafo $G(x)$ è aciclico.

3.4.5 Il sequenziamento dei lotti (BS)

Fin qui abbiamo visto come comporre i diversi lotti (*PTS*) e come sequenziare i pezzi all'interno di ciascun lotto (*IS*). Per completare il discorso, occorre specificare un modo conveniente di *sequenziare i lotti* (*BS*). L'obiettivo del problema *BS* può essere diverso asseconda delle diverse situazioni riguardanti l'attrezzaggio delle varie macchine. Ad esempio, diversi part type possono richiedere insiemi di utensili totalmente disgiunti, e quindi possiamo essere interessati a minimizzare il numero totale di operazioni di riattrezzaggio. Vale a dire, vogliamo fare in modo, quanto più possibile, che allorché viene caricato un part type, questo viene processato finché tutti i pezzi di quel tipo che abbiamo deciso di produrre nel periodo corrente sono esauriti. Nel seguito, indichiamo con (i,k) un lotto formato dai part type $i \quad P_1$ e $k \quad P_2$; sia $\{x^*\}$ una soluzione ottima di base del problema *PL1*.

Il Teorema 3.6 stabilisce che $G(x^*)$ è una foresta. Dal punto di vista della minimizzazione del numero totale di riattrezzaggi, una soluzione particolarmente favorevole si ha quando $G(x^*)$ è un cammino. In tal caso, infatti, i lotti possono essere sequenziati in modo che ogni part type e il corrispondente insieme di utensili sono caricati solo una volta: basta processare i lotti secondo l'ordine degli archi lungo il cammino. Ad esempio, se uno di questi cammini è $\{1, 2, 3,\ldots, m–1, m\}$, possiamo processare i lotti come segue: iniziamo con il lotto (1,2); dopo n_1' pezzi di tipo 1, sostituiamo tale part type con il part type 3, e processiamo quindi il lotto (3,2); e così via, fino all'ultimo lotto $(m\text{-}1,m)$.

Questa proprietà favorevole si ha per grafi anche più generali dei cammini. Infatti, lo stesso procedimento può applicarsi se, per ogni componente connessa T di $G(x^*)$, esiste un cammino π (*cammino dominante*) tale che ogni nodo di T o apartiene a π, oppure è una foglia di T. Componenti connesse di questo tipo sono chiamate *bruchi* (De Simone et al. 1990). Se ad esempio $G(x^*)$ è quello di Fig.3.25, un sequenziamento con le caratteristiche desiderate è: (1,4), (2,4), (3,4), (5,4), (5,6), (7,6), (8,6), (9,6), (9,10), (9,11).

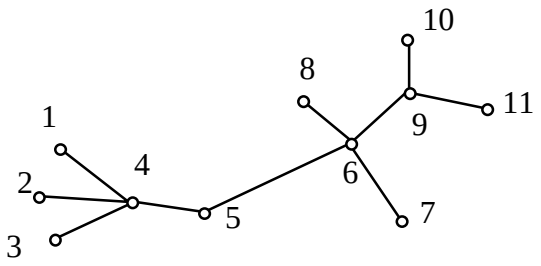


Figura 3.25. Grafo $G(x^*)$ in forma di bruco.

Se le componenti della foresta $G(x^*)$ non sono tutte bruchi (il cammino è evidentemente un perticolare bruco), possiamo avere la necessità di caricare qualche part type più di una volta. Tuttavia, le foglie di $G(x^*)$ corrispondono ancora a part type (e utensili) che saranno caricati una volta sola. Consideriamo allora la foresta $F(x^*)$ che si ottiene da $G(x^*)$ sfrondandolo di tutte le foglie, e sia π un cammino su $F(x^*)$. Dalla discussione precedente, il problema *BS* può formularsi come segue:

Sequenziamento dei lotti (BS):

Data

una soluzione ottima x^* per la formulazione PL1

trovare

una partizione $\Pi = \{\pi_1, \ldots, \pi_m\}$ degli archi di $F(x^*)$ in cammini disgiunti

tale che

il numero dei cammini sia minimo.

Infatti, il numero di cammini in Π meno il numero di componenti connesse di $F(x^*)$ è pari al numero totale di volte che accade che un part type viene caricato più d'una volta. Una partizione ottima Π può essere trovata semplicemente considerando di volta in volta, in modo greedy, cammini che uniscano due foglie (Agnetis, Arbib e Stecke 1991*a*). E' facile vedere che, se indichiamo con ω il numero di nodi di $F(x^*)$ di grado dispari, allora $|?| = \omega/2$.

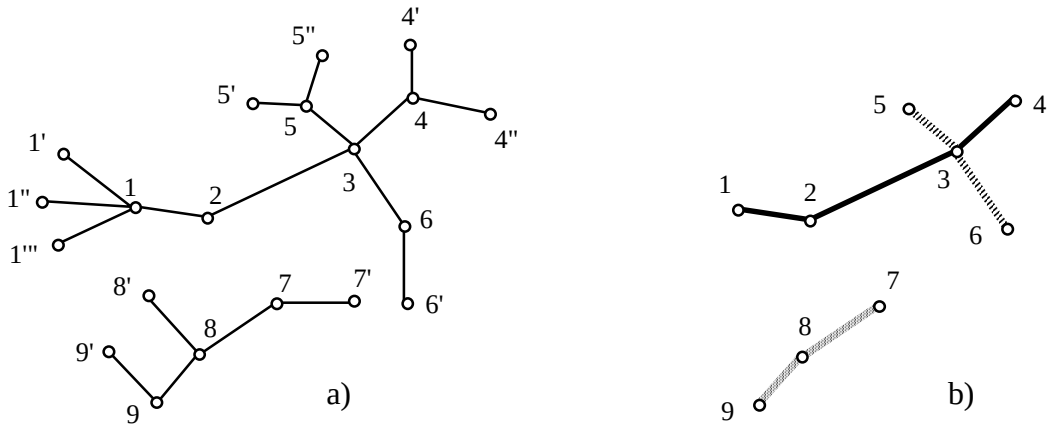


Figura 3.26.*a*) $G(x^*)$ ottenuto dalla soluzione x^* di *PL1*; *b*) La foresta $F(x^*)$, e una partizione dei suoi archi nel numero minimo di cammini.

Ad esempio, consideriamo il grafo $G(x^*)$ in Fig.3.26.*a*. Dopo avere tolto le foglie, otteniamo il grafo $F(x^*)$ in Fig.3.26.*b*, dove è anche mostrata la partizione $\Pi + \{\{1,2,3,4\}, \{5,3,6\}, \{7,8,9\}\}$ di $F(x^*)$ nel minimo numero di cammini disgiunt. Una sequenza ottima di lotti è $(1',1)$, $(1'',1)$, $(1''',1)$, $(1,2)$, $(3,2)$, $(3,4)$, $(4',4)$, $(4'',4)$, $(5',5)$, $(5'',5)$, $(3,5)$, $(3,6)$, $(6',6)$, $(7,7')$, $(7,8)$, $(8,8')$, $(8,9)$, $(9',9)$. In questa sequenza, solo il part type 3 necesita di essere caricato due volte. Infatti, si ha $|\Pi| = 3$, e $F(x^*)$ ha 2 componenti connesse.

4. TOOLING E ROUTING NEI SISTEMI FLESSIBILI DI ASSIEMATURA: PRODUZIONE SU SCALA MEDIO–GRANDE E SISTEMA DI TRASPORTO GENERALE

4.1. Introduzione

In questo capitolo viene affrontato il problema dell'instradamento con riferimento ad un sistema flessibile di assiematura costituito da un insieme di unità operatrici (*centri di lavorazione* o *macchine*) connesse da un sistema di trasporto che può avere una struttura molto generale e che comunque è in grado di collegare qualunque coppia di macchine. Le macchine sono dotate di un magazzino utensili locale, in cui la configurazione degli utensili viene determinata all'inizio di ciascun periodo di produzione e non viene variata per tutto il periodo in esame. Da questo punto di vista, la situazione è analoga a quella che si ha nella cella flessibile del §2.4. Tuttavia, questi sistemi sono sostanzialmente diversi dalle celle: oltre al fatto di avere una maggiore flessibilità di instradamento, si ha che affinché una macchina possa eseguire un'operazione su un pezzo, basta che il pezzo sia montato sulla macchina e che questa abbia l'utensile opportuno nel suo magazzino locale: non sono richieste altre risorse del sistema e dunque non sorgono problemi di sincronizzazione del tipo di quelli visti nelle celle.

Per quanto concerne il prodotto, inoltre, in questo capitolo supporremo che:

— la relazione di precedenza tra le operazioni può avere la forma di un albero generale;

— tutti i part type sono disponibili all'inizio della produzione, come ad esempio avviene allorché si presceglie un approccio di tipo *a lotti* per il problema della selezione dei part type (Stecke e Kim (1988)).

Tra i problemi di pianificazione di breve termine che possono porsi (§1.7.1), punteremo la nostra attenzione sui due più rilevanti, vale a dire l'*attrezzaggio* e l'*instradamento*. Il primo consiste nell'assegnamento di utensili ai centri di lavorazione per un periodo produttivo, per il quale è nota la quantità da produrre di ogni part type, e il secondo, che viene risolto in cascata al primo, consiste poi nell'assegnare le operazioni di ogni singolo pezzo alle varie macchine in modo da massimizzare la produttività, ovvero, come già si è avuto modo di accennare, l'obiettivo è quello (duplice) di bilanciare i carichi di lavoro e minimizzare il carico di lavoro del sistema di trasporto.

Decomporre il problema in questo modo è, se si vuole, un fatto convenzionale e non garantisce, ovviamente, l'ottimalità globale della soluzione: può darsi che con un attrezzaggio non ottimo rispetto ai criteri introdotti nel §4.3, si potrebbe ottenere una soluzione migliore, per il problema di routing, di quella che si trova col modello del §4.4. Tuttavia, la complessità del problema non sembra consentire approcci molto diversi e del resto in letteratura l'approccio di decomposizione è quello che ha ricevuto di gran lunga le maggiori attenzioni (§4.2), anche se talora la decomposizione non è tooling/routing bensì loading/scheduling (§1.6).

Un fatto distintivo dello scenario produttivo in questione è che la domanda di prodotti finiti da soddisfare nell'arco di tempo in considerazione è *medio–alta*, come del resto è tipico delle lavorazioni manufatturiere. Questa caratteristica consente di concepire algoritmi particolarmente efficienti per il problema del routing: infatti, diverse *frazioni di lotto* possono essere instradate in modo diverso, sfruttando così la flessibilità di instradamento (§1.5.1) del sistema. Questa opportunità non si ha invece se i lotti sono piccoli, allorché i vincoli di interezza non possono essere trascurati: ad esempio se solo quattro o cinque unità devono essere prodotte nel corrente periodo di pianificazione non ha senso assegnare un instradamento a 2/3 del lotto e un altro al rimanente 1/3. In questo caso, quindi, la natura intera del problema emerge inevitabilmente, ed è necessario ricorrere a formulazioni meno efficienti, quali quelle di programmazione a numeri interi; si veda in proposito (Agnetis et al. 1990).

La metodologia descritta in questo capitolo può essere applicata a fronte di diverse strategie per la selezione dei part type; comunque è particolarmente adatta quando è adottato un approccio di tipo *a lotti* (Sarin e Chen (1987), Stecke e Kim (1988)), ossia, i part type che devono essere prodotti nel prossimo periodo sono tutti determinati prima che il sistemi inizi le operazioni e l'attrezzaggio delle macchine — che costituisce la prima parte del problema — è immutato finché tutta la produzione corrente è esaurita.

Ogni unità che entra nel sistema consiste di *componenti* che devono essere lavorati e/o assiemati; la relazione di precedenza tra le operazioni è un albero–pozzo, come in Fig.4.2(a)). Ogni operazione deve essere eseguita da una macchina, attrezzata con gli utensili opportuni. L'attrezzaggio mette in condizioni ciascuna macchina di eseguire solo un sottoinsieme di tutte le operazioni. Il problema di instradamento sarà allora quello di assegnare le operazioni di ogni unità alle macchine, data la configurazione degli utensili e quindi il repertorio di operazioni che ciascuna macchina può compiere: perciò, ogni unità in generale passa attraverso varie macchine. Ogni volta che un componente o un sottoassieme viene trasferito da una

macchina all'altra, si ha un part transfer, per il quale viene impegnato il sistema di movimentazione. Il costo di un part transfer (che consiste di caricamento, spostamento e scaricamento) può dipendere fortemente dal tipo di pezzo che viene trasportato: un piccolo componente può richiedere meno tempo di un sottoassieme più grande e ingombrante. Tali costi possono entrare in modo quantitativo nella scelta degli instradamenti, come si vedrà in seguito.

Nel §4.2 faremo una breve revisione della letteratura per problemi di attrezzaggio e instradamento. Nel §4.3 sarà presentato un nuovo modello per il tooling, nel §4.4 vedremo un modello per il routing.

4.2 Letteratura sui problemi di loading in FMS e FAS

La letteratura sui problemi di attrezzaggio e instradamento è abbastanza vasta. I diversi contributi al problema si differenziano per approcci metodologici, obiettivi, struttura del sistema considerato. Come detto (§1.6), spesso in letteratura si parla di problemi di *loading*, intendendo con tale termine la decisione di allocare gli utensili alle macchine, di fatto così assegnando anche le operazioni; l'effettiva macchina su cui ciascun pezzo dovrà subire una certa operazione è in genere lasciata all'operatore, che la prenderà in tempo reale; in quest'ottica sono allora di interesse le regole di dispatching e scheduling (§1.6.2), di cui, come detto, non ci occupiamo in questo testo.

Il problema di loading è stato formulato per la prima volta come programmazione non lineare mista da Stecke (1983); successivamente Berrada e Stecke (1986) hanno messo a punto un algoritmo di branch and bound per risolvere in modo efficiente il problema, la cui non linearità risiede nei vincoli relativi alla limitata capacità dei magazzini utensili: siccome gli insiemi di utensili richiesti da ogni operazione sono in generale non disgiunti, il numero di slot occupati è minore o uguale della somma del numero di utensili richiesti da ciascun insieme. Shanker e Srinivasulu (1989) riprendono l'approccio di Berrada e Stecke e lo estendono al caso di operazioni multiple, ossia in cui vengono assegnati gruppi di operazioni a macchine. In un recente lavoro, Kouvelis e Lee (1991) propongono una diversa formulazione del problema di loading che ha una struttura particolare, che può essere risolta con un efficiente algoritmo di branch and bound. Kusiak (1984) ha proposto invece semplici modelli di programmazione intera basandosi su assunzioni semplificative. L'obiettivo è quello di bilanciare i carichi di lavoro del sistema. Tale obiettivo è stato analizzato al fine di valutare, anche attraverso modelli analitici, quanto fosse legittimo. A tale proposito va menzionato il lavoro di Stecke e Morin (1985), che dimostrano come bilanciare i carichi di lavoro in un FMS (rappresentato

tramite una rete di code chiusa), massimizza il valore atteso del throughput; l'ipotesi sul sistema è che esista un'adeguata capacità di immagazzinamento e che ogni operazione è assegnata solo a una macchina. Relativamente però a FMS con macchine raggruppate (*pooled systems*), Stecke e Solberg (1981,1985) hanno mostrato che è più conveniente assegnare un carico di lavoro maggiore alle macchine appartenenti ai gruppi più grandi, e successivamente Stecke e Kim (1989) hanno confermato il risultato attraverso delle simulazioni.

La programmazione a numeri interi è stata lo strumento più utilizzato nei problemi di loading; tra gli altri vanno ancora citati Chakravarty e Shtub (1984), Rajagopalan (1986), Stecke (1986), che usa un modello aggregato di reti di code per fornire l'input al modello di programmazione a numeri interi, Bastos (1988), che formula come programmazione mista il problema di batching (ovvero, selezione dei part type) e di loading. Sarin e Chen (1987) pongono il problema in termini di minimizzazione dei costi totali, laddove viene considerato il costo di far eseguire una particolare operazione a una macchina con un certo utensile (ad esempio, tale costo può essere legato al tempo che quella macchina impiega a eseguire l'operazione con quell'utensile). Il problema viene ripreso e risolto tramite branch and bound da Ram, Sarin e Chen (1990), utilizzando come strumento di rappresentazione del problema le reti di flusso generalizzate (ossia con dei moltiplicatori che possono "amplificare" il flusso lungo singoli archi). Wilhelm and Shin (1985) studiano il benefico effetto che può avere la duplicazione di alcuni utensili sul bilanciamento dei carichi di lavoro; non forniscono modelli esatti, bensì solo euristiche; il problema è qualitativamente simile a quello affrontato nel §4.3. Di tipo diverso, infine, è l'approccio *concorrente* di Lee e Mirchandani, che mira a risolvere contemporaneamente tooling, routing e scheduling. In alcuni casi strutturalmente semplici, vengono analizzate alcune regole che generalizzano risultati della teoria dello shop scheduling.

A causa dell'intrinseca complessità computazionale di questi approcci "esatti", si è andata formando parallelamente una vasta letteratura di metodi euristici per il loading. Una collezione di euristiche per il loading è data da Stecke e Talbot (1984), come pure da Ammons, Lofgren e McGinnis (1985), e da Chang et al. (1985). Un'interessante analisi dei vari approcci euristici è stata fatta da de Werra (1987).

Il bilanciamento dei carichi di lavoro, d'altra parte, non è l'unico obiettivo d'interesse, come già si è avuto modo di sottolineare. Lee e Jung (1990) sviluppano un approccio di goal programming, mentre Askin e Chen (1990) valutano diverse euristiche in base a diversi obiettivi.

Fin qui, si è parlato del problema del loading, in quanto molti autori non hanno affrontato in modo distinto il problema di attrezzare i centri e poi di instradare i pezzi. Kiran (1986) distingue il tooling dal routing, e dimostra la NP–completezza di varie formulazioni di questi problemi. Interessante è il lavoro di Hwang e Shanthikumar (1987) che pongono l'accento su due problemi: selezione dei part type e tooling. A fronte di una simulazione del sistema, propongono di duplicare l'utensile "bottleneck", ossia quello la cui duplicazione consente il miglior livellamento dei carichi di lavoro tra le macchine. Si noti che questa filosofia è abbastanza simile a quella che verrà utilizzata nella funzione obiettivo del problema di tooling nel §4.3.3. La convenienza di utilizzare un approccio di decomposizione, ossia di separare il tooling dal routing, è anche sostenuta da Hutchinson et al. (1991), i quali confrontano, per mezzo di simulazioni, le soluzioni ottenute tramite metodi off–line (quali ad esempio quelli presentati in questo testo) con quelle ottenibili dall'uso di regole euristiche di dispatching e scheduling. La superiorità del primo approccio risulta in molti contesti diversi. Simili conclusioni sono tratte da Yamamoto e Nof (1985). Con strumenti analoghi, Shanker e Tzen (1985) tornano a ribadire la necessità di formulare i modelli di programmazione matematica rispetto all'obiettivo di bilanciare i carichi di lavoro delle macchine. Infine, Arbib, Lucertini e Nicolò (1991) propongono, per alberi di precedenza del solo tipo *catena*, un procedimento euristico per il routing che assegna dapprima i carichi di lavoro alle macchine, e quindi sceglie un insieme di instradamenti cercando di limitare il numero di part transfer. Il loro approccio è stato perfezionato ed esteso ad alberi di assiematura qualsiasi da Agnetis e Signoretti (1991).

4.3. Configurazione degli utensili sui centri di lavorazione

4.3.1 Introduzione

In questo capitolo vedremo un modello per il problema dell'attrezzaggio in un sistema flessibile di lavorazione. In tutto il §4.3, la struttura della relazione di precedenza tra le operazioni non gioca un ruolo fondamentale, come avverrà invece nel §4.4: la trattazione cioè è valida qualunque sia la struttura dell'albero di assiematura.

Il problema di determinare il miglior modo di attrezzare i centri può essere affrontato con approcci modellistici diversi, e diverse funzioni obiettivo. Innanzitutto si osservi che esso è un problema squisitamente combinatorio, dal momento che le scelte riguardano il fatto di allocare o meno certi utensili in certi magazzini. Inoltre, l'obiettivo dell'attrezzaggio è talora di non facile identificazione, in quanto la produttività del sistema dipende dall'attrezzaggio in modo non immediato. In questo capitolo vedremo come sia possibile orientare le scelte di attrezzaggio facendo uso di

due semplici modelli di PL, in cui la struttura dei vincoli è la stessa, ma le funzioni obiettivo esprimono due diverse esigenze, e precisamente:

i) poter successivamente, in sede di routing, ottenere un buon grado di bilanciamento del carico di lavoro tra le macchine;

ii) poter successivamente instradare i pezzi avendo a disposizione un maggior numero di possibili instradamenti.

Queste due esigenze possono essere trattate indipendentemente, e allora danno luogo a due funzioni obiettivo diverse; queste, peraltro, come in ogni problema di ottimizzazione multiobiettivo, possono essere considerate insieme, facendone una combinazione convessa. Nel §4.3.2 introduciamo alcune notazioni, nel §4.3.3 gli obiettivi vengono espressi quantitativamente, e infine nel §4.3.4 vedremo come sia possibile risolvere i problemi attraverso una formulazione di costo su reti di flusso con funzione obiettivo lineare.

La trattazione che segue è analoga a quella in (Sodhi, Agnetis e Askin 1991), laddove il problema veniva decomposto in due parti da risolversi successivamente. Nel seguito, si è preferito accentrare l'attenzione sui soli modelli di PL, modificandoli leggermente in modo da rappresentare l'intero problema di attrezzaggio.

4.3.2 I problemi di attrezzaggio

Nel seguito affrontiamo il problema di decidere il modo migliore di riempire i magazzini utensili di un insieme di M macchine, a fronte di una richiesta produttiva per il prossimo periodo, costituita da P part type, per ognuno dei quali è specificata una domanda d_p (p=1,…,P). Ogni macchina m può ospitare Q_m utensili nel proprio magazzino, e supponiamo che tutti gli utensili occupino lo stesso spazio (uno slot). Ad ogni utensile u_j (j=1,…,n) è associato un tempo T_j complessivo, che tiene conto cioè anche dell'entità della domanda produttiva in quel periodo. Ogni pezzo di tipo p impiega l'utensile u_j in n_{pj} operazioni. Indichiamo con n_j la somma $\Sigma_p\, n_{pj}$.

Le M macchine che compongono il sistema flessibile possono essere uguali o diverse. In generale, non è vero che ciascun utensile può essere ospitato indifferentemente da qualunque macchina, bensì, per ogni macchina m, esisterà un insieme di utensili $U(m)$ che possono esserle assegnati. In pratica, un motivo importante per cui un utensile u_j può non essere assegnabile a una certa macchina m, è che u_j *è già presente nel magazzino utensili di* m, e dunque, esulando da considerazioni legate alla vita degli utensili, non avrebbe senso duplicarlo nello stesso magazzino: in altri termini, i modelli che andiamo a introdurre sono adatti anche per

il solo problema di *completare* la configurazione dei magazzini utensili, già parzialmente riempiti. Indichiamo con M_j' il numero di macchine che già detengono l'utensile u_j. Infine, di ogni utensile u_j sono in generale disponibili un certo numero finito (e, in genere, assai piccolo) ψ_j di copie.

E' possibile rappresentare i dati relativi alle possibilità di assegnamento di utensili a macchine attraverso un grafo $G(N,A)$ in cui nodi e archi sono definiti come segue (Fig.4.1):

— l'insieme N di nodi consiste dei due insiemi N_1 e N_2, e due nodi ulteriori p' e p''. I nodi in N_1 corrispondono ai tipi di utensili, quelli in N_2 alle macchine. p' e p'' sono sorgente e pozzo rispettivamente.

— Gli archi sono di tre tipi: per ogni utensile u_j esiste un arco (p',j); esiste un arco da un nodo $j \in N_1$ a un nodo $m \in N_2$ se e solo se la macchina m può ospitare u_j; e infine esiste un arco (m,p'') per ogni macchina m. I tre gruppi di archi verranno indicati con A_1, A_2 e A_3 rispettivamente.

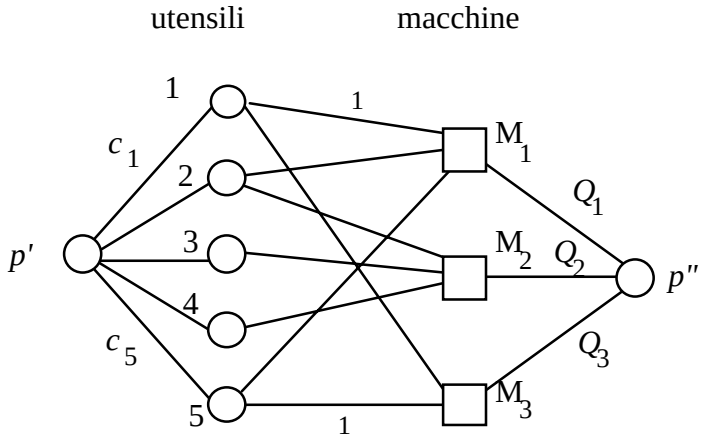


Figura 4.1. Il grafo G.

A ogni arco in G associamo un valore di *capacità* come segue:

— a ogni arco $(p',j) \in A_1$, la capacità $c_j = min\ (\psi_j, M - M_j')$;
— a ogni arco $(j,m) \in A_2$, la capacità $c_{jm}=1$;
— a ogni arco $(m,p'') \in A_3$, la capacità $c_m = Q_m$.

Il grafo G può essere interpretato allora come una *rete*, in cui il significato dei flussi nei vari archi è il seguente: il flusso in ciascun arco $(p',j) \in A_1$ rappresenta il numero totale di utensili di tipo u_j allocati nel sistema: tale valore non può eccedere il minimo tra il numero di copie disponibili di quell'utensile e il numero di macchine su cui quell'utensile può essere allocato. Il flusso in ciascun arco $(j,m) \in A_2$ rappresenta il

fatto che un utensile u_j viene aggiunto al magazzino utensili della macchina m, il che può essere solo 0 o 1. Infine, il flusso in ciascun arco (m,p'') A_3 rappresenta il numero complessivo di utensili allocati nel magazzino della macchina m, e questo dunque non può eccedere la sua capacità (ovvero la sua capacità residua, se stiamo finendo di riempire dei magazzini già parzialmente occupati).

Dunque, ad una distribuzione di flussi *intera* corrisponde un'allocazione ammissibile di utensili a macchine, ossia una soluzione del problema di attrezzaggio. Vediamo ora in base a quali obiettivi vogliamo cercare tale soluzione.

4.3.3 Bilanciamento dei carichi di lavoro per utensile

Un'idea che sembra ragionevole perseguire è che quanto più grande è il tempo di processamento associato a un utensile, tanto più è augurabile che vi siano alternative di instradamento per quell'utensile. Così, converrà "premiare" la decisione di duplicare un utensile u_j quanto maggiore è T_j. Tale criterio deve però tenere conto del vantaggio decrescente che si ha nel duplicare copie successive di uno stesso utensile. Ad esempio, consideriamo due utensili u_j e u_k, tali che T_j=100 e T_k=60. All'inizio, sono già presenti nel sistema due copie di u_j e una di u_k. Se il carico di lavoro T_j fosse equamente ripartito tra le due macchine che già possiedono u_j, ogni macchina userebbe u_j per un tempo pari a 50; invece, la sola macchina che ha u_k lo deve usare per un tempo pari a 60. Se una sola copia di un utensile può essere aggiunta, sembra più ragionevole duplicare u_k piuttosto che u_j. Allora, possiamo proporre la seguente misura di qualità per un attrezzaggio: chiamando M_j il numero di macchine che *complessivamente* (quindi, quelle che già lo avevano più quelle cui lo abbiamo aggiunto) disporranno dell'utensile u_j dopo l'attrezzaggio, possiamo essere interessati a minimizzare la somma dei rapporti tra il tempo totale di utilizzo di ciascun utensile e il numero di copie allocate alle macchine, ossia:

$$\min z_1 = \sum_{j=1}^{n} \frac{T_j}{M_j}$$

Va osservato che, in questo modo, si massimizza la quantità di tempo che può essere riallocata tra diverse macchine, il che è consistente con la definizione di flessibilità da noi data nel §1.5.2. La funzione obiettivo, in effetti, fornisce una misura di tale flessibilità.

Si noti che questa funzione obiettivo non è lineare in M_j. Occorre allora effettuare una semplice modifica alla rete G che consente di linearizzare la z_1 senza introdurre approssimazioni. Supponiamo che vi siano già presenti nel sistema 2 copie di u_j, ossia

M_j'=2. Il carico di lavoro medio per macchina relativo a u_j risulta dunque $T_j/2$. Introducendo una copia, il carico medio per macchina scende a $T_j/3$. Quindi, l'introduzione di una copia di u_j porta un beneficio pari a $T_j/6$. Si noti quindi che la funzione obiettivo z_1, ancorché non lineare, è però convessa, e questo — unitamente al fatto che tutte le soluzioni ammissibili di base di un problema di costo su reti di flusso sono intere — ci consente di risolvere il problema come programmazione lineare.

Separiamo allora ciascun arco di A_1, ossia del tipo (p',j), in c_j archi, ciascuno avente capacità unitaria, siano essi $(p',j)_1$, $(p',j)_2$,…, $(p',j)_{c_j}$. Associamo ora a ciascuno di questi archi un *costo* f_{hj}, che rappresenta il beneficio associato alla duplicazione dell'h–simo utensile u_j. Per quanto detto sopra, si ha:

$$f_{hj} = -\,[\, T_j/(M_j'+h-1) - T_j/(M_j'+h\,)\,] = -\,[\, T_j/(M_j'+h)\,(M_j'+h-1)\,]$$

tale formula vale se $M_j'+h-1>0$. Se invece $M_j'+h-1=0$, ossia se stiamo parlando dell'aggiunta della *prima* copia di un utensile che non è già presente nel sistema, il termine $T_j/(M_j'+h-1)$ va sostituito da una quantità molto elevata K_j, allo scopo di "forzare" la collocazione di almeno una copia di ciascun utensile nel sistema. Tali quantità possono essere considerate delle costanti, e quindi possono non essere considerate esplicitamente nella funzione obiettivo; tuttavia, per avere il valore corretto della somma dei carichi di lavoro medio per utensile, occorre sommare i valori K_j alla funzione obiettivo. Si noti che infatti, se vengono aggiunte h copie di u_j, il carico di lavoro medio per utensile è dato, correttamente, da $T_j/(M_j'+h)$, pari alla somma $K_j + \sum_{s=1}^{h} f_{sj}$. Infatti, dal momento che i benefici marginali sono decrescenti, verrà sempre selezionato (e saturato) prima l'arco $(p',j)_1$, quindi $(p',j)_2$ etc. Tutti gli altri archi del grafo hanno costo nullo.

Associando allora una variabile y_{hj} a ciascun arco in A_1, e una variabile x_{jm} a ciascun arco in A_2, possiamo scrivere la formulazione risultante del problema come:

$$\min z_1 = \sum_{j=1}^{n} \sum_{h=1}^{c_j} f_{hj}\, y_{hj}$$

$$\sum_{h=1}^{c_j} y_{hj} = \sum_{m=1}^{M} x_{jm} \qquad j=1,2,\ldots,n$$

$$\sum_{j=1}^{n} x_{jm} = Q_m \qquad m=1,2,\ldots,M$$

$$0 = x_{jm} = 1$$

Come già osservato, non è necessario vincolare esplicitamente all'interezza i valori delle variabili in quanto la matrice dei vincoli è totalmente unimodulare e dunque tutte le soluzioni di base, tra cui quella ottima, saranno sicuramente intere.

4.3.4 Massimizzazione del numero di instradamenti

In questo paragrafo vediamo una formulazione che differisce da quella precedente per la sola funzione obiettivo. Mentre nel §4.3.3 abbiamo considerato il carico di lavoro associato a ciascun utensile, senza preoccuparci direttamente del numero di instradamenti, stavolta vogliamo analizzare direttamente il problema di massimizzare il numero di instradamenti che un pezzo può seguire.

Se indichiamo con S_p l'insieme degli utensili richiesti da un pezzo di tipo p, supponendo che un utensile u_j sia usato n_{pj} volte per pezzo, e indicando al solito con M_j il numero di copie dell'utensile u_j presenti nel sistema, il numero di instradamenti disponibili per il part type p è dato da

$$\prod_{u_j \in S_p} M_j^{\,n_{pj}}$$

se allora volessimo massimizzare il numero complessivo di instradamenti tra tutti i part type, ecco che potremmo formulare questo obiettivo:

$$z_2' = \max \sum_{p=1}^{P} \prod_{u_j \in S_p} M_j^{\,n_{pj}}$$

a differenza però del caso precedente, z_2' non è convessa, e di conseguenza ottimizzare rispetto a questa funzione può essere estremamente più difficile, dal momento che non è più vero, in generale, che l'ottimo giace su un vertice della regione ammissibile. Tuttavia, va osservato che questo obiettivo può non essere il migliore dal punto di vista pratico. Infatti, quello che può accadere è un indesiderato "sbilanciamento" tra i numeri di instradamenti disponibili per ogni part type: supponiamo ad esempio che per tre part type il numero di instradamenti inizialmente disponibili sono 1, 8 e 24 rispettivamente. Ora, la duplicazione di un utensile usato da una sola operazione in ogni part type raddoppia il numero di instradamenti per quel part type; di conseguenza, il terzo part type sarebbe chiaramente preferito ai primi due, e quindi l'obiettivo z_2' porterebbe a trascurare il secondo e soprattutto il primo part type. Sembra allora più interessante formulare funzioni obiettivo che migliorano al crescere del numero di instradamenti, ma che li distribuiscono in modo bilanciato tra i diversi part type. Basta allora considerare il *prodotto* del numero di instradamenti

per i diversi part type, anziché la loro somma. Infatti si osservi che, a parità di numero complessivo di instradamenti, tale funzione obiettivo raggiunge il massimo quando gli instradamenti sono bilanciati il più possibile tra i diversi part type. Consideriamo perciò la seguente funzione obiettivo:

$$z_2'' = \max \prod_{p=1}^{P} \prod_{u_j \; S_p} M_j^{n_{pj}}$$

In questa forma, il problema è di programmazione non lineare. Tuttavia, si ha che

$$\log \prod_{p=1}^{P} \prod_{u_j \; S_p} M_j^{n_{pj}} = \sum_{p=1}^{P} \sum_{u_j \; S_p} \log M_j^{n_{pj}}$$

e, essendo il logaritmo una funzione monotona, possiamo in definitiva scrivere l'obiettivo come

$$z_2 = \max \sum_{j=1}^{n} n_j \log M_j$$

siccome vogliamo massimizzare una funzione concava su un insieme convesso, si può ragionare come nel §4.3.3: precisamente, esprimendo ancora il problema in forma di minimizzazione, basta associare stavolta a ciascuna variabile y_{hj} un costo f_{hj} dato da:

$$f_{hj} = -n_j (\log(M_j' + h) - \log(M_j' + h - 1))$$

come prima, infatti, la convessità della funzione obiettivo e la totale unimodularità della matrice dei coefficienti garantiscono che il problema può essere risolto come PL.

4.3.5 Risultati sperimentali

Per la valutazione della "bontà" delle soluzioni proposte, occorre risolvere il problema di instradamento e vedere quali soluzioni è stato possibile ottenere con i diversi attrezzaggi. Allo scopo è stato utilizzato da Sodhi, Agnetis e Askin (1991) un semplice modello di PL come quello usato da Wilhelm e Shin (1985) in cui ad ogni variabile di decisione corrisponde un instradamento di quelli resi possibili dalla risoluzione del problema di attrezzaggio con le due funzioni obiettivo, e l'obiettivo tiene conto della necessità di bilanciare i carichi di lavoro tra le macchine e di non appesantire il sistema di trasporto. Tale formulazione consente di trovare il routing ottimo rispetto a qualunque combinazione convessa di questi due obiettivi, tuttavia il

numero di variabili può essere esponenziale, e dunque è stato impiegato solo a fini sperimentali, mentre dal punto di vista pratico risulta molto più utile un modello del tipo di quello illustrato nel prossimo §4.4.

Le conclusioni dell'analisi possono riassumersi come segue:

— L'obiettivo z_1 ha dato risultati migliori in termini di minimizzazione del carico di lavoro della macchina più carica;

— L'obiettivo z_2 ha dato risultati migliori dal punto di vista della minimizzazione del carico di lavoro del sistema di trasporto;

— L'efficacia di ambedue le strategie aumenta all'aumentare del numero di slot vuoti che vanno riempiti;

— Quando gli slot vuoti sono pochi, z_1 dà risultati globalmente migliori, quando sono molti, z_2 è preferibile;

— Ambedue le strategie consentono nella quasi totalità dei casi di bilanciare i carichi di lavoro attorno a un valore molto prossimo al minimo teorico (somma di tutti i T_j diviso il numero di macchine).

Nel prossimo capitolo vedremo un modello di instradamento che si colloca logicamente in cascata a quelli visti in questo capitolo. D'ora in poi quindi gli utensili caricati su ogni macchina sono fissati e di conseguenza ogni macchina è abilitata a compiere un certo insieme di operazioni.

4.4 Il problema dell'instradamento nei FAS

4.4.1 Introduzione

Sulla base dell'attrezzaggio dei centri di lavorazione ottenuto ad esempio con un modello come quello del precedente §4.3, vogliamo ora risolvere il problema di instradare ciascun pezzo nel sistema.

Come visto nel §4.2, diversi obiettivi possono proporsi per il problema del routing in un sistema flessibile di produzione. Uno dei più comuni e importanti è quello di *bilanciare i carichi di lavoro* tra le macchine, che per noi qui vorrà dire minimizzare il carico di lavoro della macchina più carica. Affinché la soluzione al problema del routing dia luogo ad una distribuzione di carichi di lavoro utile, non si può però ignorare l'importanza di *limitare il carico di lavoro del sistema di movimentazione*: troppi part transfer possono sovraccaricare i sistema di trasporto che

potrebbe allora diventare il collo di bottiglia dell'intero sistema (Stecke e Solberg 1981, Agnetis et al. 1990, Arbib, Lucertini e Nicolò 1991).

In questo capitolo viene presentato un nuovo approccio al problema del routing per part type di cardinalità medio–alta, in cui viene affrontato il duplice obiettivo di bilanciare i carichi di lavoro tra le macchine e minimizzare il carico del sistema di trasporto. L'approccio consiste nella risoluzione sequenziale di due problemi di PL. Il primo è un problema di piccole dimensioni e consente di calcolare il minimo carico di lavoro della macchina più carica; il secondo invece, tra i routing che bilanciano i carichi di lavoro tra le macchine, ne determina uno che minimizza i costi complessivi di part transfer. Siccome in generale il numero delle variabili del secondo problema cresce esponenzialmente col numero di operazioni, è necessario ricorrere a tecniche particolari per risolvere il problema in modo efficiente: Agnetis e Signoretti (1991) hanno messo a punto una tecnica basata sulla generazione di colonne, in cui il problema di separazione è risolubile in tempo polinomiale.

Per semplicità, ci riferiremo in seguito a un singolo part type: la metodologia può comunque essere immediatamente estesa al processamento contemporaneo di diversi part type, come discusso in un caso reale da Agnetis e Signoretti (1991).

Il capitolo è organizzato come segue. Nel §4.4.2 il problema di instradamento viene formulato in modo preciso, nel §4.4.3 vengono descritti i modelli di programmazione lineare; e nel successivo §4.4.4 è mostrato come un routing ottimo può essere derivato dalla loro soluzione. Nel §4.4.5 è illustrata la tecnica a generazione di colonne. Nel §4.4.6 è discussa la complessità; nel §4.4.7 l'intero procedimento è illustrato con un esempio. Infine nel §4.4.8 vengono presentati brevemente i risultati relativi a prove numeriche condotte su alberi generati casualmente.

4.4.2 Formulazione del problema

Sia M l'insieme delle macchine ($|M|=m$) e N l'insieme delle operazioni che devono essere eseguite su ogni unità ($|N|=n$). Sia τ_i il tempo richiesto per eseguire l'operazione O_i su qualunque macchina (ossia, le macchine hanno la stessa velocità). Le operazioni *non* possono essere interrotte e poi riprese. Per indicare quali operazioni possono essere eseguite da ciascuna macchina, possiamo definire un grafo bipartito $B = (N, M, E)$ (Fig.4.2(*b*)), in cui

$$E + \{(i,j) \mid O_i \quad N, M_j \quad M, i \text{ può essere eseguita da } M_j\} \quad (1)$$

Sia $T=(N,A)$ l'*albero di assiematura*, in cui, al solito, i nodi corrispondono alle operazioni e un'operazione O_k non può iniziare prima che siano terminate tutte le operazioni O_i tali che (i,k) A. Per semplicità, supponiamo di numerare topologicamente i nodi di T, di modo che se (i,k) A, allora $i < k$. Quindi, la radice di T ha l'etichetta n. Il successore di i verrà indicato come $\sigma(i)$. In Fig.4.2 è mostrato un esmpio, con n=9 e m=5. Il prodotto consiste di 4 componenti (infatti, l'albero ha 4 foglie).

Il problema consiste nel *trovare, per ogni unità che entra nel sistema, un assegnamento di operazioni a macchine*. Ad esempio, la Fig.4.3(*b*) mostra un assegnamento in cui alla macchina M_1 sono assegnate le operazioni O_1, O_2, O_3 e O_9; a M_2 le operazioni O_5 e O_8; a M_3 le operazioni O_6 e O_7 e infine a M_4 l'operazione O_4. La figura 4.3(a) illustra la partizione di T indotta dall'assegnamento: possiamo riconoscere 4 *part transfers*, precisamente quelli tra le operazioni O_3 e O_5 (infatti, O_3 è assegnato a M_1 e O_5 a M_2, e dunque un sottoassieme deve essere trasportato da M_1 a M_2), O_4 e O_5, O_7 e O_9, O_8 e O_9. Possiamo associare a ogni nodo i N il *costo* c_i del part transfer che si ha se O_i e il suo successore $O_{\sigma(i)}$ sono eseguiti da macchine diverse. Tale costo può tener conto della complessità di operazioni di caricamento e scaricamento, e questo a sua volta può dipendere dalla forma e dall'ingombro del sottoassieme che viene trasportato. Il trasferimento del prodotto finito fuori dal sistema deve essere compiuto indipendentemente dagli assegnamenti prescelti, perciò non può essere evitato e possiamo assumere il suo costo c_n=0.

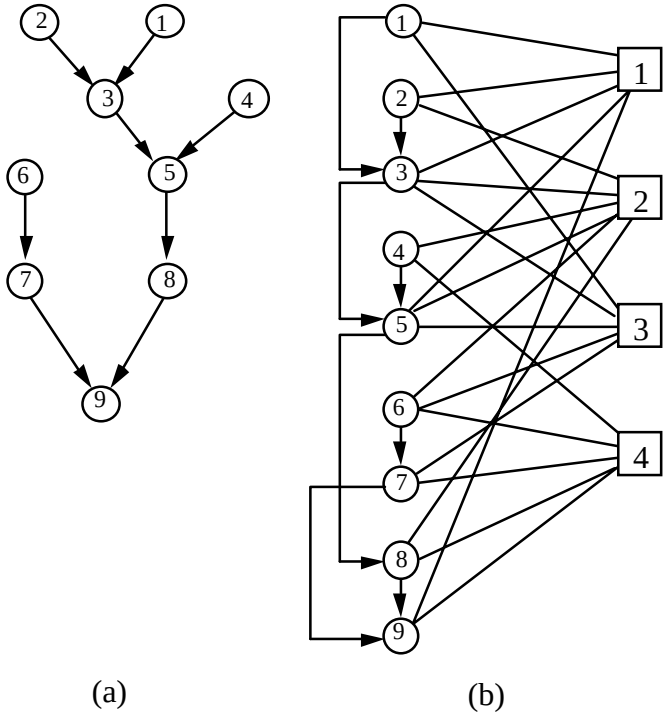


Figura 4.2. (a) Albero di assiematura T (b) Grafo bipartito B.

Ci occuperemo degli obiettivi di bilanciare i carichi di lavoro tra le macchine e minimizzare il costo complessivo di movimentazione pezzi; se tutti i part transfer hanno lo stesso costo il secondo obiettivo corrisponde, evidentemente, a minimizzare il *numero* complessivo di part transfer.

Come in ogni problema di ottimizzazione multiobiettivo, ci sono vari modi di combinarli. L'approccio considerato consiste nel *minimizzare i costi totali di part transfer, tra tutte le soluzioni che bilanciano i carichi di lavoro in modo ottimo*. Il problema può quindi formularsi come segue:

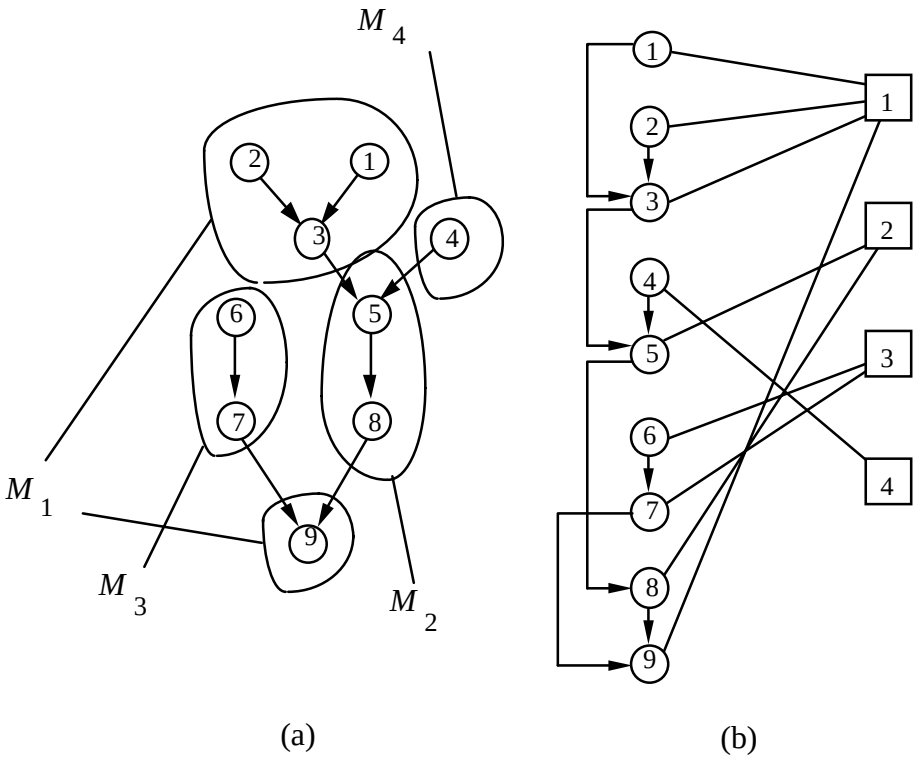


Figura 4.3. (a) Partizione dell'albero di assiematura (b) Assegnamento corrispondente.

***Problema di Routing (ROP)**:*

Dato
un insieme di lotti, ciascuno caratterizzato da un albero di assiematura T; e un attrezzaggio delle macchine, espresso attraverso il grafo bipartito B;

trovare
un instradamento che bilancia i carichi di lavoro in modo ottimo

tale che
il costo complessivo di tutti i part transfer sia minimo.

Il minimo valore z^* della macchina più carica è calcolato per mezzo di un problema di PL, indicato con *WBAL*; quindi, viene risolto un altro PL, in cui si minimizza il costo complessivo di instradamento, col vincolo che il carico di lavoro di ciascuna macchina non ecceda z^* (questo secondo problema sarà indicato come *PT*).

4.4.3 Formulazione dei problemi come PL

4.4.3.1 *Minimizzazione del carico di lavoro z* della macchina più carica*

Nel problema *WBAL*, sia x_{ij} la *frazione di lotto* per la quale l'operazione O_i è eseguita dalla macchina M_j. Chiaramente, $0 = x_{ij} = 1$.

(*WBAL*) $\quad min \quad z$

$$\text{s.t.} \quad \sum_{j:\,(i;j)\ E} x_{ij} = 1 \qquad i = 1, \ldots, n \qquad (2)$$

$$\sum_{i:\,(i;j)\ E} \tau_i\, x_{ij} = z \qquad j = 1, \ldots, m \qquad (3)$$

$$x_{ij} = 0 \qquad \forall\ (i,j) \quad E \qquad (4)$$

I vincoli (2) assicurano che, per ogni unità del lotto, l'operazione O_i è assegnata a qualche macchina; i vincoli (3) esprimono il carico di lavoro di ciascuna macchina; l'obiettivo è quello di minimizzare il massimo tra questi carichi di lavoro. Il valore ottimo z^* è un parametro del prossimo problema *PT*.

4.4.3.2 *Minimizzazione dei costi di part transfer*

Prima di formulare il problema *PT*, dobbiamo introdurre alcune definizioni. Dato il grafo *B*, consideriamo la coppia $<T_h,M_j>$, in cui T_h è un sottoalbero di *T* tale che tutte le sue operazioni possono essere eseguite dalla macchina M_j. Chiamiamo *modulo* questa coppia e sia *Q* l'insieme di tutti i moduli ($|Q|=q$). Nel seguito, r_h indicherà il nodo radice del sottoalbero T_h. Si noti che diversi moduli possono essere definiti dallo stesso sottoalbero: nell'esempio di Fig.4.2(b), $T_h+\{O_6,O_7\}$ è presente in ambedue i moduli $<T_h,M_3>$ e $<T_h,M_4>$. Possiamo associare a ogni modulo $Q_{hj}+<T_h,M_j>$ un *costo* c_{r_h}, dato dal costo associato alla radice di T_h (indipendentemente dalla macchina M_j). Sia θ_h la somma dei tempi operazione τ_i, con $O_i \ T_h$. Diciamo che un nodo *i precede un sottoalbero* T_h se *i* è il predecessore di un nodo di T_h e *i* non appartiene a T_h.

Chiaramente, un assegnamento definisce una partizione in moduli dell'albero di assiematura *T*. Ad esempio, in Fig.4.3 è raffigurato un assegnamento che utilizza 5 moduli: $M = \{<\{1,2,3\},M_1>, <\{4\},M_4>, <\{5,8\},M_2>, <\{6,7\},M_3>, <\{9\},M_1>\}$. Perciò, a un routing corrisponde l'insieme di moduli impiegati in tutti gli assegnamenti. Il costo di part transfer complessivo di un routing è chiaramente dato dalla somma dei costi di tutti i moduli selezionati. In conclusione, il problema consiste allora nel trovare, per ogni unità, un insieme di moduli tali che il costo complessivo sia minimizzato.

Il problema può essere formulato come il seguente PL, in cui y_{hj} è la frazione di lotto che utilizza il modulo Q_{hj}; a_{ih} è 1 se il sottoalbero T_h contiene l'operazione O_i e 0 altrimenti.

(*PT*) $$\min \zeta = \sum_{hj:\, Q_{hj} \in Q} c_{r_h} y_{hj}$$

$$\sum_{hj:\, Q_{hj} \in Q} a_{ih}\, y_{hj} = 1 \qquad i=1,2,\ldots,n \qquad (5)$$

$$\sum_{h:\, Q_{hj} \in Q} \theta_h\, y_{hj} = z^* \qquad j=1,2,\ldots,m \qquad (6)$$

$$y_{hj} = 0 \qquad h,j:\, Q_{hj} \notin Q \qquad (7)$$

I vincoli (5) hanno lo stesso significato dei vincoli (2) in *WBAL*; i vincoli (6) impediscono al carico di lavoro di ciascuna macchina di eccedere il valore z^*, assicurando così l'ottimalità dal punto di vista del bilanciamento dei carichi.

Sia $\boldsymbol{y}^*$ la soluzione ottima di *PT*: y_{hj}^* è dunque il valore della frazione di lotto assegnata al modulo $<T_h, M_j>$ nella soluzione ottima di *PT*. Se indichiamo con *r* la cardinalità del lotto, il numero di unità che usano il modulo Q_{hj} è ovviamente ry_{hj}^*. Tale quantità può non essere intera, e viene così introdotto un errore di arrotondamento; tuttavia, quanto più grande è *r*, tanto più piccolo è l'errore di approssimazione (vedi anche (Agnetis et al. 1990)).

4.4.4 Calcolo del routing

La soluzione al problema *PT* non specifica direttamente un routing che utilizza questi moduli. Tuttavia, esso può essere facilmente ottenuto, calcolando gli assegnamenti e le frazioni di lotto ad essi associate, per mezzo dell'approccio greedy descritto di seguito.

Anzitutto, calcoliamo il primo assegnamento $M^{[1]}$. Si consideri la radice dell'albero di assiematura. Ovviamente, deve esistere almeno un modulo $<T_h, M_j>$ contenente la radice (O_9 nell'esempio di Fig.4.2) e tale che $y_{hj}^*>0$ (ciò è assicurato dai vincoli (5) di *PT*). Assegnamo allora tutte le operazioni di T_h a M_j. Proseguiamo quindi nella costruzione dell'assegnamento: per ogni nodo *i* che precede il sottoalbero T_h dobbiamo considerare un modulo $<T_k, M_l>$ tale che $y_{kl}^*>0$ con T_k avente radice in

i; si può facilmente verificare che per ogni *i* che precede T_h ne esiste sempre almeno uno. Questo modo di scegliere i moduli prosegue verso i livelli superiori dell'albero finché tutte le operazioni sono state assegnate. A questo punto, calcoliamo la frazione di lotto $f^{[1]}$ associata a questo assegnamento: essa è determinata dal minimo y_{hj}* con Q_{hj} $M^{[1]}$. Quindi, aggiorniamo i valori ottimi y_{hj}* associati ai moduli di $M^{[1]}$ sottraendo da essi il valore $f^{[1]}$, dopodiché si ricomincia il calcolo di un altro assegnamento, $M^{[2]}$. L'algoritmo va avanti finché l'intero lotto è instradato (ossia, ***y****=**0**). In conclusione, dalla soluzione ottima del problema *PT* un routing ottimo può ottenersi per mezzo del seguente semplice algoritmo:

Algorithm_ROUTING (**input**: soluzione ottima di *PT* ***y****, albero di assiematura *T*=(*N*,*A*);

output: insieme *M* di assegnamenti e frazioni associate);

begin

M := Ø; *k*:=0;

{inizializzazione}

While ***y**** ? **0 do** {calcola un nuovo assegnamento $M^{[k]}$}

begin

k:=*k*+1;

$M^{[k]}$:= Ø; $f^{[k]}$:= 0;

R:={*n*};

While *R*?Ø **do**

begin

scegli un nodo *i* *R* e cancellalo da *R*;

seleziona un modulo Q_{hj} +<T_h, M_j> con radice in *i* e tale che y_{hj}*>0;

$M^{[k]}$:= $M^{[k]}$≈ Q_{hj} ;

aggiungi a *R* tutti i nodi che precedono T_h;

end*;*

$f^{[k]}$:=min{y_{hj}*| Q_{hj} $M^{[k]}$} {calcola la frazione di lotto associata a $M^{[k]}$} **for all** Q_{hj} $M^{[k]}$ **do** y_{hj}*:=y_{hj}*–$f^{[k]}$; {aggiorna i valori y_{hj}*}

M := *M* ≈ $M^{[k]}$

end;

end.

Si noti che, in generale, la scelta del routing non è unica. In definitiva, *ROP* può essere risolto attraverso la seguente procedura:

Algorithm SOLVE_ROP (**input**: albero di assiematura $T=(N,A,\boldsymbol{\tau},\boldsymbol{c})$; grafo bipartito $B=(N,M,E)$;
output: insieme M di assegnamenti e frazioni di lotto associate);

begin

Risolvi *WBAL* ottenendo il minimo carico di lavoro di bottleneck z^*;
Risolvi *PT* con parametro z^* e ottieni i valori ottimi $\boldsymbol{y^*}$;
ROUTING ($\boldsymbol{y^*}$, $T=(N,A)$; M);

end.

Nel prossimo paragrafo accenniamo a come risolvere *PT* in modo efficiente.

4.4.5 Una tecnica a generazione di colonne per risolvere PT

Il numero totale q di moduli che occorre considerare nella formulazione di *PT* dipende fortemente dalla struttura dell'albero di assiematura. In molte applicazioni, l'albero di assiematura è una catena: ad esempio, quando i sottoassiemi devono subire una *serie* di lavorazioni, come del resto già visto nel §2. In tal caso, il numero di sottoalberi connessi è $O(n^2)$, e quindi il numero di moduli è alpiù $O(mn^2)$. D'altra parte, se T consiste di una *singola* operazione di assiematura per mezzo della quale $(n-1)$ components sono messi insieme, il numero di sottoalberi connessi è $O(2^{n-1})$ e perciò dovremmo considerare, in generale, $O(m\,2^{n-1})$ moduli. Di conseguenza, in alcuni casi può essere praticabile, dal punto di vista computazionale, generare *tutte* le colonne di *PT*, ma in molti casi no. E' però sempre possibile risolvere *PT* in tempo polinomiale, tramite un approccio a generazione di colonne.

Consideriamo il duale di *PT*. Associamo a ciascuna operazione una variabile v_i e a ogni macchina una variabile u_j :

$$(D) \qquad \min \eta = \sum_{i=1}^{n} v_i + z^* \sum_{j=1}^{m} u_j$$

$$\sum_{i=1}^{n} a_{ih}\, v_i + \theta_h\, u_j = -c_h \qquad h,j:\ Q_{hj} \in Q \qquad (8)$$

$$u_j = 0 \qquad j=1,2,\ldots,m \qquad (9)$$

$$v_i \quad \text{non vincolato} \qquad i=1,2,\ldots,n \qquad (10)$$

Per risolvere *PT*, quello che si fa è considerare una versione "ridotta" del problema primale, in cui cioè compare solo un numero ristretto di variabili. Tale problema ristretto sarà indicato come *RP*, mentre *RD* si riferisce al suo duale. Sia Q_R l'insieme dei moduli corrispondenti alle variabili in *RP*; siano $\boldsymbol{y_R^*}$ e ($\boldsymbol{u_R^*}$, $\boldsymbol{v_R^*}$) le soluzioni ottime di *RP*e *RD*rispettivamente. L'insieme iniziale delle variabili di *RP*può essere scelto in molti modi diversi; una possibilità è quella di selezionare tutti i *singletons* (ossia, moduli Q_{hj}+<T_h, M_j> in cui T_h consiste di una sola operazione) e tutti i *moduli*

massimali (ossia, moduli Q_{hj}+<T_h, M_j> tali che T_h non è contenuto in alcun altro sottoalbero interamente eseguibile da M_j). Tale scelta garantisce l'esistenza di una soluzione ammissibile di *RP*; in questo modo inizialmente $|Q_R|=O(mn)$. Si consideri ora una soluzione ottima $\boldsymbol{y_R^*}$ di *RP*, e sia $\boldsymbol{y'}$ la soluzione a *PT* ottenuta come segue:

$$y'_{hj} = y_{Rhj}^* \quad \text{se } Q_{hj} \quad Q_R$$
$$y'_{hj} = 0 \quad \text{se } Q_{hj} \quad Q_R$$

Uno di questi due casi può presentarsi: $\boldsymbol{y'}$ è ottima per *PT* (e quindi $\{\boldsymbol{u_R^*}, \boldsymbol{v_R^*}\}$ per (*D*)), oppure no (e quindi $\{\boldsymbol{u_R^*}, \boldsymbol{v_R^*}\}$ è inammissibile per (*D*)). In quest'ultimo caso, nasce allora il problema di *trovare uno o più vincoli di D violati da* $\{\boldsymbol{u_R^*}, \boldsymbol{v_R^*}\}$ (problema di separazione). Se vengono individuati uno o più vincoli di *D*, le corrispondenti variabili del primale vengono aggiunte all'insieme Q_R, dopodiché si risolve nuovamente *RP*. La procedura prosegue finché si arriva a un punto in cui non esistono più vincoli violati e quindi si è individuata una soluzione ottima per *PT*. Il problema di separazione, per *ROP*, equivale a cercare il sottoalbero connesso di peso minimo su un albero i cui nodi hanno pesi positivi e negativi; esso può essere risolto facilmente in tempo lineare (Agnetis e Signoretti 1991).

4.4.6 Complessità computazionale dell'algoritmo

Vogliamo ora analizzare la complessità computazionale dell'approccio risolutivo proposto. L'intero algoritmo consiste di tre blocchi: risoluzione di *WBAL*, risoluzione di *PT*, calcolo dell'insieme di assegnamenti.

WBAL è un PL di piccole dimensioni, con $O(mn)$ variabili.

PT è risolto con una tecnica a generazione di colonne a ogni passo della quale viene risolto un problema *RP* e un'istanza del problema di separazione, il quale è risolubile in tempo $O(mn)$. A ogni passo possono essere individuati $O(mn)$ vincoli violati, ed è perciò del tutto praticabile aggiungere tutti i corrispondenti moduli a Q_R.

Per quanto concerne il calcolo del routing, osserviamo che ogni assegnamento può essere calcolato in tempo $O(n)$. Il numero totale di moduli <T_h, M_j> tali che y_{hj}*>0 non può superare $m+n$ (numero di vincoli di *PT*). Quindi, dalla soluzione $\boldsymbol{y^*}$ di *PT*, il routing può essere calcolato in tempo $O((m+n)n)$.

Osserviamo che siccome il problema di separazione è risolto in tempo polinomiale, il problema *ROP* è risolubile in tempo polinomiale (Grötschel, Lovasz e Schrijver (1981)).

Da un punto di vista pratico, il modo in cui viene scelto l'insieme iniziale di moduli in *RP* dà luogo a un numero molto piccolo di problemi di PL che devono essere risolti per arrivare alla soluzione ottima di *PT*. I risultati delle prove numeriche sono illustrati nel §4.4.8.

4.4.7 Esempio

Al fine di illustrare l'applicazione del metodo proposto, analizziamo il seguente esempio. Si consideri l'albero di assiematura in Fig.4.2(a) e il grafo bipartito in Fig.4.2(b). Il sistema consiste di 4 macchine. I dati relativi all'esempio sono riassunti in Tav.1, che indica i tempi di processamento e i costi di part transfer associati a ciascuna operazione.

Tavola 4.1. Dati relativi all'esempio.

i	1	2	3	4	5	6	7	8	9
τ_i	5	5	12	4	15	4	8	12	15
c_i	1	1	2	1	3	1	1	3	0

Risolvendo *WBAL* si ottiene il valore ottimo z^*=20 del carico di lavoro della macchina più carica. Si noti che in questo esempio esso eguaglia il valore minimo teorico ($\Sigma_i\ \tau_i / m$). Quindi, risolviamo *PT* per trovare il routing che minimizza i costi di part transfer, col vincolo che il carico di lavoro di nessuna macchina superi z^*=20.

Il primo passo della procedura iterativa proposta consiste nel selezionare un numero limitato di variabil iniziali in *RP*. Scegliendo, per ogni macchina M_j, le variabili corrispondenti a tutti i singletons e ai *sottoalberi massimali*, al primo passo del nostro approccio venivano selezionate 26 variabili; la soluzione dell'istanza iniziale di *RP* dà, come costo di trasporto complessivo, il valore 5.3895.

Quindi, si analizza la soluzione ottima di *RD* alla ricerca di qualche *vincolo violato* di (*D*). L'algoritmo di separazione rivela che è violato il vincolo di *D* relativo al modulo $<\{O_8, O_9\}, M_4>$:

$$v_8 + v_9 + 27u_4 = 0.$$

Successivamente, l'algoritmo di separazione seleziona i seguenti due moduli:

$$< \{3,2,1\}, M_1>$$
$$< \{3,1\}\ \ , M_3>$$

Le 3 corrispondenti variabili sono aggiunte a *RP*, che viene risolto nuovamente. Risolvendo poi ancora il problema di separazione, è facile verificare che stavolta nessun vincolo di *D* è violato, e quindi l'attuale soluzione a *RP*specifica una soluzione ottima per il problema *PT*.

Nella soluzione ottima di *PT*, dieci variabili sono non nulle, tutte corrispondenti a sottoalberi distinti. I dieci moduli Q_{hj} sono elencati nella Tav.4.2, con il corrispondente valore di y_{hj}* arrotondato alla seconda cifra decimale. La prima colonna della Tav.4.2 indica il "tipo" di modulo Q_{hj}: S sta per singleton, M per sottoalbero massimale rispetto a M_j, N per *nuovo* modulo, aggiunto nel corso dell'algoritmo. La somma dei valori sulla sesta colonna dà il valore della funzione obiettivo, 5.05.

A questo punto, ci rimane solo da calcolare un insieme di assegnamenti facenti uso dei moduli nella soluzione ottima di *PT*. Come descritto nel §4.4.3, dalla Tav.4.2

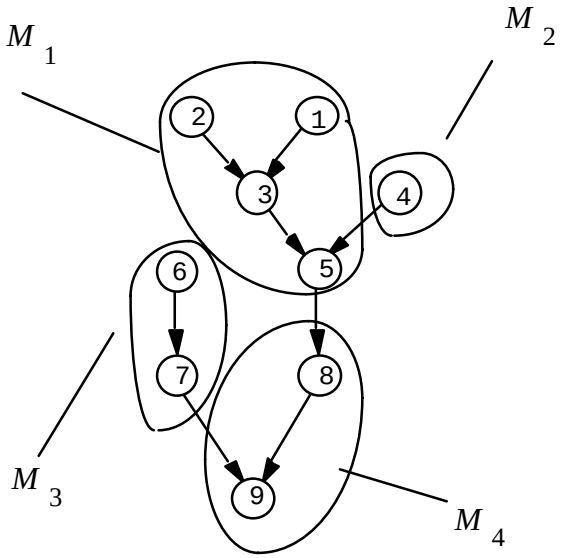
selezioniamo un modulo contenente la radice (O_9): ad esempio, scegliamo il modulo $<T_1+\{9,8\}, M_4>$, per il quale $y_{1,4}^*=0.41$ (numeriamo i sottoalberi asseconda dell'indice della corrispondente riga nella Tav.4.2). Così, assegnamo, O_8 e O_9 a M_4. Dopodiché, dobbiamo considerare moduli con radice in O_7 e O_5; scegliamo quindi $<T_5+\{7,6\}, M_3>$, per cui $y_{5,3}^*=0.77$, e $<T_6+\{5,3,2,1\}, M_1>$, per cui $y_{6,1}^*=0.36$. La sola operazione rimasta è O_4, e il solo modulo che ha radice in tale nodo è $<T_8+\{4\}, M_2>$, per il quale $y_{8,2}^*=0.64$. L'assegnamento definito da questi moduli è mostrato in Fig.4.4(a). La corrispondente frazione di lotto $f^{[1]}$ è ottenuta come minimo tra tutti i valori y_{hj}^*; perciò $f^{[1]}=\min\{0.41, 0.77, 0.36, 0.64\} = 0.36$. I quattro valori y_{hj}^* associati ai moduli selezionati sono quindi diminuiti della quantità $f^{[1]}$; questo dà luogo ai nuovi valori $y_{hj}^{*[1]}$ indicati nella quarta colonna di Tav.4.3. Si noti che adesso $y_{6,1}^{*[1]}=0$. Con lo stesso procedimento, si sceglie quindi un nuovo assegnamento, e così via. In Tavola 5 sono mostrati i valori delle variabili $y_{hj}^{*[k]}$ dopo k assegnamenti; un routing completo è specificato dai quattro assegnamenti, illustrati in Fig.5.

Tavola 4.2. Soluzione ottima di *PT* nell'esempio.

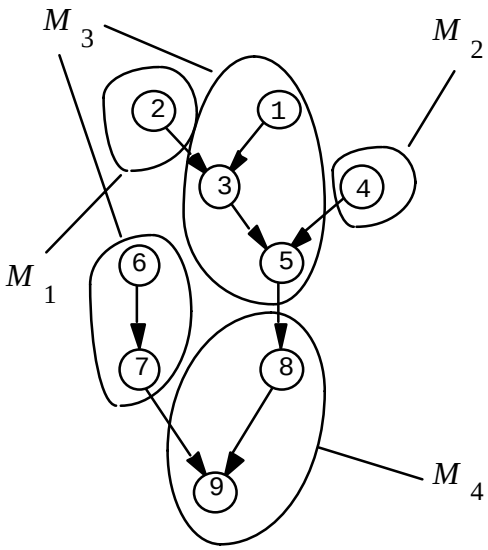
h	*Type*	M_j	T_h	y_{hj}^*	c_{r_h}	$c_{r_h} y_{hj}^*$	θ_h	$r\,\theta_h y_{hj}^*$
1	N	4	9,8	0.41	0	0.00	27	1107
2	S,M	1	9	0.36	0	0.00	15	540
3	M	4	9,8,7,6	0.23	0	0.00	39	897
4	M	2	8,5,4,3,2	0.36	3	1.08	48	1728
5	M	3	7,6	0.77	1	0.77	12	924
6	M	1	5,3,2,1	0.36	3	1.08	37	1332
7	M	3	5,3,1	0.28	3	0.84	32	896
8	S	2	4	0.64	1	0.64	4	256
9	S	1	2	0.28	1	0.28	5	140
10	S	3	1	0.36	1	0.36	5	180

Tavola 4.3. Valori delle variabili $y_{hj}^{*[k]}$.

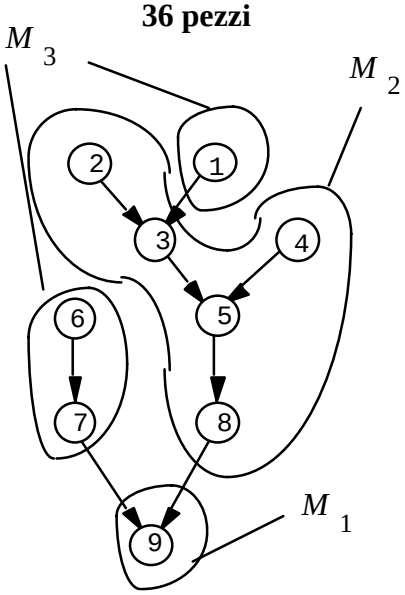
h	M_j	T_h	$y_{hj}^{*}[0]$	$y_{hj}^{*}[1]$	$y_{hj}^{*}[2]$	$y_{hj}^{*}[3]$
1	4	9,8.	**0.41**	**0.05**	0	0
2	1	9.	0.36	0.36	**0.36**	0
3	4	9,8,7,6.	0.23	0.23	0.23	**0.23**
4	2	8,5,4,3,2.	0.36	0.36	**0.36**	0
5	3	7,6.	**0.77**	**0.41**	**0.36**	0
6	1	5,3,2,1.	**0.36**	0	0	0
7	3	5,3,1.	0.28	**0.28**	0.23	**0.23**
8	2	4.	**0.64**	**0.28**	0.23	**0.23**
9	1	2.	0.28	**0.28**	0.23	**0.23**
10	3	1.	0.36	0.36	**0.36**	0



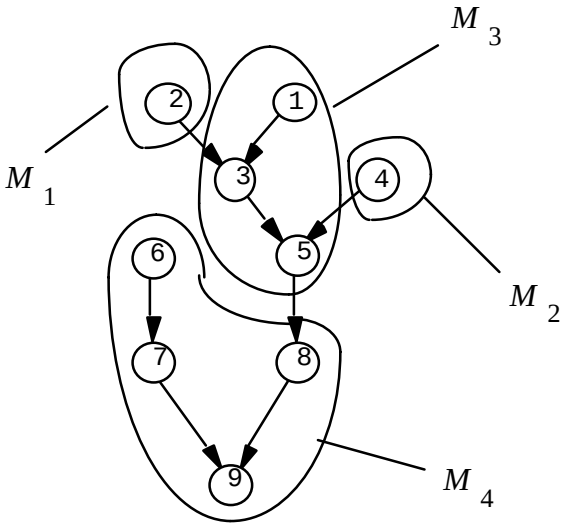
36 pezzi



5 pezzi



36 pezzi



23 pezzi

Figura 4.4. Soluzione del problema con r=100 unità.

4.4.8 Prove numeriche

Al fine di valutare l'efficienza del metodo proposto, sono state condotte alcune prove numeriche su alberi di assiematura generati casualmente. Gli alberi considerati sono alberi binari completi, consistenti di 31 operazioni. Supponiamo che m=5. Tutti i costi c_h sono identici, quindi la funzione obiettivo PT rappresenta il numero totale di part transfers. Tutte le operazioni si suppone abbiano la stessa durata.

Sono stati sviluppati molti esempi random, caratterizzati da diverse strutture del grafo bipartito B. In altre parole, solo i tipi di operazioni eseguibili su ogni macchina variano. Abbiamo generato 100 esempi, in cui ogni macchina poteva eseguire circa 50% di tutte le operazioni — col vincolo che ogni operazione doveva essere eseguibile da almeno una macchina.

I risultati degli esperimenti sono riassunti in Tav.4.4. Sono riportate informazioni relative al numero di operazioni di pivot e di problemi di separazione risolti. Osserviamo che l'approccio a generazione di colonne ha portato un incremento molto basso nel tempo di calcolo rispetto alla soluzione dell'istanza iniziale di RP: l'aumento medio nel numero di operazioni di pivot è inferiore al 30% e in media occorre risolvere solo tre o quattro problemi di separazione. Il numero di variabili non cresce in modo significativo: in genere solo il 12% del numero iniziale di variabili sono aggiunte nel corso dell'algoritmo; tale valore non ha mai ecceduto 30%.

Questi risultati consentono di concludere che l'approccio sviluppato in questo capitolo è assolutamente compatibile con le risorse di calcolo disponibili in tutti i casi pratici.

Tavola 4.4. Risultati delle prove numeriche.

Valori medi	prima della generaz. di colonne	dopo la generaz. di colonne
no. di variabili	67.05	75.44
no. di operazioni di pivot	66.97	86.27
no. di problemi di separazione risolti		3.72
no. di var. aggiunte a ogni separazione		2.25
diminuzione nella funzione obiettivo		8.06 %